



Technisch-Naturwissenschaftliche  
Fakultät

# Programmieraufgaben für Pidgets und den Raspberry Pi im Informatikunterricht

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Magister der Naturwissenschaften

im Diplomstudium

LEHRAMT INFORMATIK UND INFORMATIKMANAGEMENT /  
PHYSIK

Eingereicht von:  
Klaus Rabeder

Angefertigt am:  
Institut für Systemsoftware

Beurteilung:  
a.Univ.-Prof. Dipl.-Ing. Dr. Günther Blaschek

Linz, Juli 2013

# Inhaltsverzeichnis

1	Einleitung .....	5
1.1	Aufgabenstellung .....	5
1.2	Motivation .....	5
1.3	Voraussetzungen .....	6
1.4	Didaktische Überlegungen .....	6
1.5	Überblick über diese Arbeit .....	7
2	Der Raspberry Pi .....	8
2.1	Raspberry Pi Modelle .....	9
2.2	Zusatzkomponenten für den Betrieb .....	9
2.3	Mögliche Arbeitsszenarien im Unterricht .....	10
2.3.1	Raspberry Pi als PC .....	10
2.3.2	Raspberry Pi im WLAN, Zugriff über XTerminal .....	11
2.3.3	Raspberry Pi im LAN, Zugriff über XTerminal .....	12
3	Software Installation .....	13
3.1	Übersicht Betriebssysteme .....	13
3.2	Betriebssystem installieren – Windows .....	14
3.3	Betriebssystem installieren – Mac OS X .....	14
3.4	Grundeinstellungen .....	16
3.5	Netzwerkeinstellungen .....	17
3.6	Paketverwaltung installieren .....	18
3.7	Software aktualisieren .....	19
3.8	Zusatzpakete installieren .....	20
3.9	IDE Geany mit root-Privilegien ausführen .....	21
3.10	Pythoncompiler ändern .....	22
3.11	Benutzer hinzufügen .....	22
3.12	Hilfsprogramme .....	23
3.12.1	Hilfsprogramme Windows .....	24
3.12.2	Hilfsprogramme für Mac OS X .....	28
4	Pidget Klassifikation .....	31
4.1	Interaktion mit der Umwelt .....	31
4.2	Charakter .....	31
4.3	Fachzugehörigkeit .....	31
4.4	Kommunikation .....	32
4.5	Technische Realisierbarkeit .....	32
4.6	Verständlichkeit der Funktionsweise .....	32
4.7	Stromverbrauch .....	33
4.8	Preis .....	33

5	Realisierte Pidgets .....	34
5.1	Kriterien .....	34
5.1.1	Interaktion mit der Umwelt .....	34
5.1.2	Charakter .....	34
5.1.3	Fachzugehörigkeit .....	34
5.1.4	Kommunikation .....	35
5.1.5	Technische Realisierbarkeit .....	35
5.1.6	Verständlichkeit der Funktionsweise .....	35
5.2	Technische Grundlagen zum GPIO Port .....	35
5.2.1	Elektrische Grenzwerte .....	37
5.2.2	Initialzustand der GPIO Pins .....	37
5.2.3	Designhinweise .....	38
5.3	Dokumentation Pidgets .....	39
5.3.1	Pi-Car .....	40
5.3.2	LED Strip .....	42
5.3.3	7-Segment Anzeige .....	43
5.3.4	Optical Transmitter .....	45
5.3.5	Optical Receiver .....	47
5.3.6	Digital servo .....	48
5.4	Bauanleitung Dice Pidget und Verbindungskabel .....	49
5.4.1	Schaltplan und Funktion .....	50
5.4.2	Benötigte Bauteile und Werkzeuge .....	51
5.4.3	Arbeitsschritte .....	51
5.4.4	Verbindungskabel .....	53
6	Pidget-Module .....	55
6.1	Dice .....	55
6.2	LED Strip .....	56
6.3	7-Segment Display .....	58
6.4	Optical Transmitter .....	59
6.5	Optical Receiver .....	60
7	Programmieraufgaben .....	62
7.1	Lauflicht .....	62
7.1.1	Lehrerinformation .....	62
7.1.2	Arbeitsblatt .....	64
7.1.3	Musterlösung .....	66
7.1.4	Musterantworten .....	67
7.2	Binärzahlendarstellung .....	68
7.2.1	Lehrerinformation .....	68
7.2.2	Arbeitsblatt .....	69
7.2.3	Musterlösungen .....	70

7.2.4	Musterantworten.....	70
7.3	Binary Battle .....	71
7.3.1	Lehrerinformation .....	71
7.3.2	Arbeitsblatt .....	73
7.3.3	Musterlösung.....	75
7.3.4	Musterantworten.....	76
7.4	Black Jack .....	77
7.4.1	Lehrerinformation .....	77
7.4.2	Arbeitsblatt.....	79
7.4.3	Musterlösungen .....	82
7.4.4	Musterantworten.....	84
7.5	Clock .....	86
7.5.1	Lehrerinformation .....	86
7.5.2	Arbeitsblatt.....	87
7.5.3	Musterlösungen .....	89
7.5.4	Musterantworten.....	90
7.6	Morse Transmitter.....	91
7.6.1	Lehrerinformation .....	91
7.6.2	Arbeitsblatt.....	93
7.6.3	Musterlösungen .....	95
7.6.4	Musterantworten.....	96
7.7	Morse Receiver .....	97
7.7.1	Lehrerinformation .....	97
7.7.2	Arbeitsblatt.....	99
7.7.3	Musterlösung.....	104
7.7.4	Musterantworten.....	106
8	Rück- und Ausblick.....	107
9	Quellenverzeichnis .....	108
10	Abbildungsverzeichnis .....	111
11	Tabellen.....	112
12	Listings .....	113
13	Danksagungen .....	114
14	Lebenslauf .....	115
15	Eidesstattliche Erklärung.....	116

# 1 Einleitung

Der Raspberry Pi ist ein Einplatinencomputer in der Größe einer Kreditkarte, der von der Raspberry Pi Foundation entwickelt wurde und um ca. 35€ verkauft wird. Die Motivation für die Entwicklung des Geräts war, dass die Entwickler, die unter anderem vom University of Cambridge's Computer Laboratory kommen, bemerkt haben, dass nicht nur die Anzahl, sondern auch die Fähigkeiten der Bewerber für das Informatikstudium kontinuierlich sinkt bzw. sinken. Waren in den 1990er Jahren die meisten Bewerber noch Hobbyprogrammierer, so hatten sich im Jahr 2000 die meisten Bewerber höchstens mit ein wenig Web-Design beschäftigt. Das Ziel war, mit dem Raspberry Pi ein Gerät zu entwickeln, bei dem man sich (ähnlich wie früher bei Home Computern) nach dem Startvorgang in einer Programmierumgebung befindet. [1]

Der Begriff Pidgets ist eine Wortschöpfung des Autors und setzt sich aus Pi (von Raspberry Pi) und Widgets („Vorrichtung“ oder umgangssprachlich „Dingsbums“) zusammen. Es handelt sich dabei um kleine Hardware-Bausteine, die an die GPIO-Schnittstelle (General Purpose Input/Output) des Raspberry Pi angeschlossen werden und über Leuchtdioden, Taster und ähnliche Elemente sowohl mit der Umwelt als auch mit dem Raspberry Pi kommunizieren.

## 1.1 Aufgabenstellung

In dieser Diplomarbeit sollen Pidgets entwickelt werden, die für Programmieraufgaben im Informatik-Unterricht an Schulen eingesetzt werden können. Die Pidgets sollen einfach aufgebaut sein, damit sie mit geringen Kosten leicht nachgebaut werden können.

Zu den Pidgets sollen Aufgabenstellungen entwickelt werden, in denen die Schüler<sup>1</sup> – aufbauend auf eine vorangegangene Einführung in das Programmieren – Handlungskompetenzen in den Bereichen Anwenden und Gestalten erlangen können.

Weiters sollen für die Aufgabenstellungen Arbeitsblätter, Musterlösungen und Begleitmaterial für Lehrkräfte erstellt werden. Als Programmiersprache soll Python verwendet werden. [2]

## 1.2 Motivation

Die Idee für diese Arbeit ist, dass mit der Kombination aus Pidgets und dem Raspberry Pi etwas geschaffen wird, bei dem die Programmierung „den Computer verlässt“ und mehr mit der realen Welt interagiert, als dies andere Aufgabenstellungen im Informatik-Unterricht üblicherweise tun.

Auf Grund der geringen Kosten des Raspberry Pi, eignet sich das Gerät hervorragend als „Bastel-Rechner“, und das in zweifachem Sinn: einerseits können die Schüler in einer Umgebung, „in der sie nichts kaputt machen können“ in der Programmierung alles ausprobieren (was ja eine der Intentionen der Entwickler ist). Andererseits birgt die Verbindung von Computer und selbst gebauter Elektronik zwar die Gefahr, dass Teile davon zerstört werden, der finanzielle Schaden ist jedoch im Vergleich zum potentiellen Gewinn vernachlässigbar. Dieser Gewinn liegt nach Sichtweise des Autors darin, dass der experimentelle Charakter der Kombination Raspberry Pi plus Pidgets die Phantasie der Schüler anregt und zusätzlich die Wissensvermittlung besser greifbar macht. Es zeigen viele der Projekte, die seit Veröffentli-

---

<sup>1</sup> Es sei an diese Stelle ausdrücklich betont, dass mit dem Begriff „Schüler“ die Gesamtheit aller Schülerinnen und Schüler gemeint ist. Genauso verhält es sich mit „Benutzer“ oder „Lehrer“. Es wurde in dieser Arbeit zu Gunsten der besseren Lesbarkeit darauf verzichtet, jeweils beide Geschlechter dieser Personengruppen anzuschreiben. Eine diskriminierende Absicht ist damit nicht verbunden.

chung des Raspberry Pi weltweit realisiert wurden, dass die GPIO Schnittstelle des Raspberry Pi die Möglichkeit zu einer neuen Klasse von Anwendungen eröffnet. Als Zusatznutzen wird noch das Verständnis gefördert, aus welchen Teilen ein Computer besteht und wie diese Teile miteinander interagieren.

### 1.3 Voraussetzungen

Diese Arbeit setzt voraus, dass die Schüler in die Programmierung eingeführt wurden und dass grundsätzliche Dinge, wie Kontrollstrukturen, Datentypen, Methodenaufrufe, etc. bekannt sind. Ein empfehlenswertes online Tutorial für die in dieser Arbeit verwendete Programmiersprache Python bietet Codecademy. [3]

Das Pidget darf für die Schüler keine Black Box sein – weder von der Programmierung, noch von der elektronischen Funktionsweise. Die Idee ist, dass die Ergebnisse einer Programmierung in der realen Welt der Schüler ankommen. Daher ist es der Idealfall, wenn die Schüler ihre Pidgets selber aufbauen und ihnen anschließend über die selbstgeschriebenen Programme „Leben einhauchen“. Die elektronischen Schaltungen der Pidgets sind bewusst einfach gehalten, damit für die Schüler nicht ein weiteres „abstraktes Ding“ ins Spiel kommt. Der Physiker James Clerk Maxwell (1831-1879) hat es so formuliert: *"The educational value of such experiments is often inversely proportional to the complexity of the apparatus."* [4]

Seitens der Lehrer ist es wünschenswert, dass sie ein ausreichendes elektronisches Grundverständnis mitbringen und es schaffen, den Unterricht so zu gestalten, dass die Arbeit mit den Pidgets in gleicher Weise als lehrreich und unterhaltend empfunden wird.

### 1.4 Didaktische Überlegungen

Die Programmieraufgaben, die für diese Arbeit entwickelt werden, sind so zu formulieren, dass sie an eine Einführung in das Programmieren aus der 5. Klasse im Wahlpflichtgegenstand Informatik anknüpfen. Die Schüler sind also grundsätzlich an Informatik interessiert, haben aber auf Grund der vielen anderen Themen, die in der 5. Klasse behandelt werden müssen, erst wenig selbst programmiert.

Ein Ziel der Aufgaben ist es, die Schüler bei ihrem Wissensstand abzuholen und ihnen durch eine Heranführung an die informatische Denkweise das Rüstzeug für größere und komplexere Aufgaben mitzugeben. Ein weiteres Ziel ist durch häufige Teamarbeit die Fähigkeit der Schüler zur Zusammenarbeit zu fördern.

Die Arbeitsblätter werden so gestaltet, dass die Schüler bei der Bearbeitung der Aufgaben gut angeleitet werden und sollen zwei Funktionen erfüllen: zuerst sollen sich die Schüler erarbeiten, wie die Aufgabenstellung modelliert werden kann. Dann soll die Implementierung in kleinen und überschaubaren Teilaufgaben erfolgen.

Die Unterrichtsformen werden so gewählt, dass es immer wieder Möglichkeiten gibt, die Schüler auf einen gemeinsamen Wissensstand und Fertigstellungsgrad der Aufgabe zu bringen. Für leistungsstärkere Schüler werden Bonusaufgaben angeboten.

Der programmiertechnische Umfang der Aufgaben soll mit maximal 100 Zeilen Code sehr überschaubar bleiben. Trotzdem soll jede gelöste Aufgabe das verwendete Pidget in ein fertiges Gerät oder Spiel verwandeln. Für die Ansteuerung der Pidgets werden Python-Module angeboten, die für jedes Pidget eine Anzahl von Methoden zur Verfügung stellen.

Für höhere Klassen oder Klassen mit umfangreicheren Vorkenntnissen kann der Schwierigkeitsgrad der Aufgaben erhöht werden, in dem die zur Verfügung gestellten Hilfsmittel reduziert werden. Dies kann bis zur Reduktion der Angabe auf eine Software-Spezifikation gehen, oder dass die Schüler anstatt der angebotenen Python-Module, die Ansteuerung der Pidgets selbst programmieren.

Aufgrund der variierbaren Schwierigkeitsgrade der Aufgaben einerseits und weil der Lehrplan für den Wahlpflichtgegenstand (WPG) Informatik ein Rahmenlehrplan ohne konkrete Verknüpfung der Lehrinhalte mit der Schulstufe ist, können die einzelnen Aufgaben keiner bestimmten Schulstufe zugeordnet werden.

## 1.5 Überblick über diese Arbeit

Am Ende der Einleitung soll noch ein Überblick über die weiteren Kapitel dieser Arbeit gegeben werden.

In Kapitel 2 werden der Raspberry Pi und seine Ein- und Ausgänge beschrieben, sowie die verschiedenen Modelle verglichen. Es folgt eine Aufstellung von Zusatzkomponenten, die für den Betrieb benötigt werden und eine Gegenüberstellung von verschiedenen möglichen Arbeitsszenarien im Unterricht. Dieses Kapitel kann Entscheidungshilfen geben, wenn der Ankauf von Raspberry Pi Computern für eine ganze Informatikgruppe erwogen wird und es ermöglicht auch eine Abschätzung der zu erwartenden Gesamtkosten.

Kapitel 3 beschreibt die Installation der Software für den Raspberry Pi. Zuerst erfolgt eine Übersicht über verfügbare Betriebssysteme, dann werden die Betriebssysteminstallation und das Durchführen von Grundeinstellungen beschrieben. Ein weiterer Abschnitt widmet sich der Software, die die Schüler für ihre Arbeit benötigen, und schließlich wird das Hinzufügen von weiteren Benutzern gezeigt. Am Ende des Kapitels werden für die Betriebssysteme Windows und Mac OS X Programme vorgestellt, die für die Erstinstallation des Betriebssystems am Raspberry Pi oder für verschiedene Arbeitsszenarien im Unterricht benötigt werden.

In Kapitel 4 wird eine Möglichkeit zur Klassifizierung der Pidgets gezeigt. Dazu werden die Merkmale eines Pidgets und deren mögliche Ausprägungen angeführt und mit Beispielen veranschaulicht.

In Kapitel 5 werden die realisierten Pidgets dokumentiert. Zunächst werden die in Kapitel 4 eingeführten Merkmale hinsichtlich ihrer Eignung für den Einsatz im Unterricht betrachtet. Diese Betrachtung liefert die Entscheidungsgrundlage für die realisierten Pidgets. Im Anschluss folgen technische Grundlagen zum GPIO. Im vorletzten Teil des Kapitels erfolgt die Dokumentation der realisierten Pidgets: verbale Beschreibung, Klassifikation nach Kapitel 4, Schaltplan, Stückliste und der Materialpreis. Zum Abschluss des Kapitels wird exemplarisch an einem Pidget gezeigt, wie dieses Schritt-für-Schritt aufgebaut werden kann.

In Kapitel 6 befindet sich für jedes Pidget die Beschreibung und der Quellcode für ein Python-Modul, das die Schüler in ihre Programme importieren können. Mit Hilfe dieser Module können die Pidget-Funktionen über Methodenaufrufe genutzt werden.

In Kapitel 7 sind die Programmieraufgaben dokumentiert. Jede Aufgabe besteht aus einer Lehrerinformation mit didaktischen und pädagogischen Aspekten zur Aufgabe, einem Arbeitsblatt mit den Aufgabenstellungen und den Bonusaufgaben, einer Musterlösung in Form des Quellcodes und Musterantworten auf die Fragen im Arbeitsblatt.

Den Abschluss bildet Kapitel 8 mit einem Rück- und Ausblick.

## 2 Der Raspberry Pi

Dieses Kapitel widmet sich dem Herzstück der Arbeit – dem Raspberry Pi. Der Raspberry Pi ist ein Einplatinencomputer (vgl. Abbildung 1), der in etwa die Größe einer Scheckkarte hat (Abmessungen der Platine: 85 x 55 mm). Der Name des Computers ist phonetisch gleich dem englischen Wort für Himbeerkuchen (*Raspberry Pie*). Ursprünglich sollte der Computer mit fest eingebautem Interpreter für die Programmiersprache Python geliefert werden, deshalb steht das „Pi“ im Namen für „Python Interpreter“. Die „Himbeere“ knüpft an die Tradition an, Computer nach Früchten zu benennen (wie z.B. Apple, Apricot, Acorn). [5]

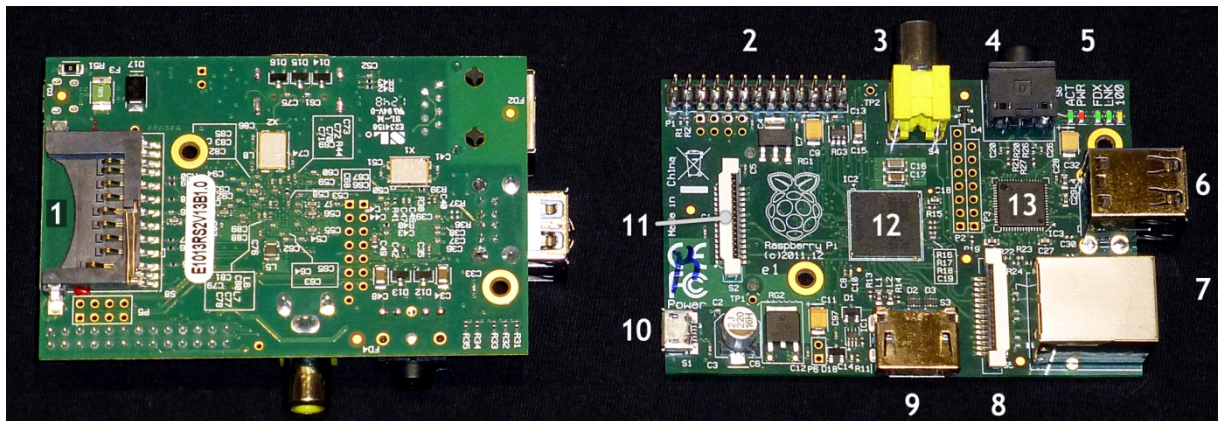


Abbildung 1: Vorder- und Rückseite des Raspberry Pi

Auf der Platine befinden sich folgende Anschlüsse und Komponenten:

Nr	Bezeichnung	Beschreibung
1	SD-Kartenslot	Steckplatz für die SD-Karte (enthält Betriebssystem des Raspberry Pi)
2	General Purpose Input/Output (GPIO) Port	Diese Anschlüsse können als Ausgang oder als Eingang konfiguriert werden. An diesem Port werden die Pidgets angeschlossen
3	Composite Videoausgang	Anschluss für ein analoges TV-Geräte oder einen Monitore
4	Audioausgang	Das Audiosignal wird vom Raspberry Pi entweder auf diesem Anschluss (analog) oder am HDMI Ausgang (digital) ausgegeben
5	Status LEDs	ACT (grün)      Zugriff auf SD-Karte PWR (rot)        Betriebsspannung liegt an FDX (grün)      full duplex LAN Verbindung LNK (grün)      Ethernetverbindung / LAN Aktivitätsanzeige 100 (gelb)      10 oder 100Mbit LAN Verbindung
6	2x USB	Zwei Universal Serial Bus (USB) 2.0 Anschlüsse für Peripheriegeräte
7	Netzwerkanschluss	Ethernet Anschluss
8	Camera Serial Interface (CSI)	Anschluss für eine Kamera
9	HDMI Ausgang	Anschluss für ein digitales TV-Gerät oder einen PC Monitor. Dieser Ausgang kann auch das Audiosignal übertragen
10	Micro USB Eingang	5V Spannungsversorgung des Raspberry Pi
11	Display Serial Interface (DSI)	Anschluss für ein Display
12	Broadcom BCM2835	Ein-Chip-System (SoC). Enthält CPU, GPU, DSP und RAM
13	LAN Controller	Chip für die Netzwerkschnittstelle

Tabelle 1: Anschlüsse und Komponenten des Raspberry Pi



## 2.1 Raspberry Pi Modelle

Zum Zeitpunkt der Erstellung dieser Arbeit hatte die Raspberry Pi Foundation drei Versionen des Raspberry Pi entwickelt – es sind dies Modell A, Modell B und Modell B2. Tabelle 3 zeigt Gemeinsamkeiten und Unterschiede der drei Modelle. [5], [6]

	Modell A	Modell B	Modell B2
SoC	Broadcom BCM2835		
CPU	700MHz AMR1176JZF-S Core (AMR11)		
GPU	Broadcom VideoCore IV, OpenGL ES2.0, 1080p30h.264/MPEG-4-AVC-Decoder		
Speicher	256MB		512MB
USB Anschlüsse	1	2 (über integrierten Hub)	
Videoausgang	Composite RCA, HDMI (rev. 1.3 und 1.4) Auflösung von 640x350 bis 1920x1200 Pixel		
Netzwerkanschluss	Keiner	10/100Mbit Ethernet	
Stromaufnahme	500mA (2,5W)	700mA (3,5W)	
Preis [7]	23,40 €	Nicht verfügbar	32,88 €

Tabelle 2: Vergleich Raspberry Pi Modelle

Die Modelle A bzw. B und B2 können aufgrund des (nicht) vorhandenen Netzwerkanschlusses unterschieden werden. Eine genaue Information, um welche Revision des Raspberry Pi es sich handelt, liefert der folgende Konsolenbefehl:

```
cat /proc/cpuinfo
```

Dieser Befehl liefert eine Textausgabe. Der Eintrag in der Zeile „Revision“ ermöglicht eine genaue Zuordnung des Raspberry Pi anhand Tabelle 3. [8]

Revision	Veröffentlichungsdatum	Modell	Speicher
Beta	Q1 2012	B (Beta)	256MB
0002	Q1 2012	B	256MB
0003	Q3 2012	B (ECN0001)	256MB
0004	Q3 2012	B	256MB
0005	Q4 2012	B	256MB
0006	Q4 2012	B	256MB
0007	Q1 2013	A	256MB
0008	Q1 2013	A	256MB
0009	Q1 2013	A	256MB
000d	Q4 2012	B2	512MB
000e	Q4 2012	B2	512MB
000f	Q4 2012	B2	512MB

Tabelle 3: Zuordnung Revision zu Modell

### Hinweis:

- Falls vor der Revisionsnummer die Zahl 1000 steht (z.B. 10000009), dann bedeutet das, dass der Prozessor schon einmal übertaktet wurde. Weitere Informationen zum Übertakten finden sich in Kapitel 3.4 Grundeinstellungen.

## 2.2 Zusatzkomponenten für den Betrieb

Um den Raspberry Pi betreiben zu können, sind zusätzliche Komponenten notwendig, die in Tabelle 4 samt Bezugsquelle und Preis aufgelistet sind. Je nach gewähltem Arbeitsszenario (vgl. Kapitel 2.3) wird nur ein Teil der aufgelisteten Komponenten benötigt.

Zubehör	Beispiel	Zweck	Quelle	Preis in €
Tastatur, Maus USB	MS Wired Desktop 400 bulk	Benutzereingaben	e-tec.at	12,85
Tastatur, Maus Funk	Logitech K260	Benutzereingaben	e-tec.at	21,90
Monitor (DVI)	ASUS VW199NR, 19"	Bildschirmausgabe	conrad.at	99,95
HDMI auf DVI Adapter	Adapter - HDMI Stecker auf DVI-D Buchse	Verbindung zw. RPi und DVI-Monitorkabel	amazon.at	0,99
Monitor (HDMI)	AOC e2262Vwh, 21.5"	Bildschirmausgabe	amazon.at	112,92
HDMI Kabel	Kabel HDMI High Speed 1,0m	Verbindung zw. RPi und Monitor	netzwerkshop.at	1,56
Netzteil 5V	Hama USB-Ladegerät 5V/2.1A	Spannungsversorgung	amazon.at	13,10
Micro-USB Kabel	Kabel USB-A Stecker an Micro-B Stecker, 1m	Verbindung Netzteil - RPi	amazon.at	3,39
Speicherkarte	Transcend SDHC 16GB Class 10	Nicht flüchtiger Massenspeicher	amazon.at	12,21
Kartenleser	Transcend USB Kartenlesegerät	Beschreiben/Sichern der SD-Karte	amazon.at	5,90
Netzwerkkabel	CAT.6 Patchkabel 3m	Netzwerkverbindung	amazon.at	2,60
Switch	Netzwerk Switch 16x 10/100MBit/s	Vernetzen mehrerer RPi	amazon.at	22,79
USB WLAN Adapter	EDIMAX EW-7811UN Wireless USB Adapter	Drahtlose Netzwerkverbindung	amazon.at	10,70
WLAN Router	Netgear RangeMax N150	Vernetzen mehrerer RPi	amazon.at	21,90

Tabelle 4: Zusatzkomponenten

**Hinweis:**

- Bei der Auswahl des 5V Netzteils ist zu beachten, dass dieses auch ausreichend Strom liefern kann. Das Modell B2 des Raspberry Pi verbraucht 700mA. Dazu addieren sich der Stromverbrauch von Peripheriegeräten (Tastatur, Maus, USB WLAN Adapter, etc.) und der Stromverbrauch des angeschlossenen Pidgets. Es wird daher empfohlen, ein Netzteil zu wählen, das 1,5 – 2A liefern kann. In Leistung ausgedrückt entspricht das 7,5 – 10W.

Unterdimensionierte Netzteile führen zu Schwankungen der Versorgungsspannung und in der Folge zu Instabilitäten des Raspberry Pi.

**2.3 Mögliche Arbeitsszenarien im Unterricht**

Im letzten Abschnitt dieses Kapitels werden verschiedene Kombinationen aus Raspberry Pi, Zubehörkomponenten und ggf. Software beschrieben, die Kosten pro Arbeitsplatz berechnet und das jeweilige Szenario auf Vor- und Nachteile im Unterricht untersucht. Es wird dabei davon ausgegangen, dass es für den Informatikunterricht einen EDV-Raum gibt in dem für jede Schülerin und jeden Schüler ein Windows-PC bereitsteht. Diese PCs sind vernetzt, haben eine Verbindung zum Internet und sind zumindest „mittelmäßig“ am Stand der Technik: also Tastatur und Maus mit USB Anschluss, Monitor mit DVI- oder HDMI-Anschluss und als Betriebssystem mindestens Windows XP.

**2.3.1 Raspberry Pi als PC**

In diesem Szenario werden an den Raspberry Pi die vorhandenen Peripheriegeräte (Maus, Tastatur, Monitor) und das vorhandene Netzwerkkabel angeschlossen. Für den Betrieb des Raspberry Pi sind somit nur mehr folgende Komponenten notwendig:

Gerät	Preis in €
Raspberry Pi (Modell B2)	32,88
SD-Karte	12,21
HDMI auf DVI Adapter	0,99
Micro-USB Kabel	3,39
Netzteil	13,10
<b>Gesamtkosten/Arbeitsplatz</b>	<b>62,57</b>

Tabelle 5: Kosten bei Betrieb als PC

Für die Schüler ist dieser Aufbau einfach zu überschauen und zu verstehen – ein Standard-PC wird durch den Raspberry Pi ersetzt. Die Peripheriegeräte bleiben die gleichen, nur der Bildschirminhalt ändert sich naturgemäß, weil am Raspberry Pi ein anderes Betriebssystem läuft. Um mit der Arbeit am Raspberry Pi beginnen zu können benötigen die Schüler noch eine Einführung in den Linux Desktop, die Entwicklungsumgebung und Informationen über das Dateisystem (wo sind die Angaben zu finden, wo müssen die Lösungen gespeichert werden).

Der Nachteil dieses Szenarios ist der relativ hohe zeitliche Aufwand, den Raspberry Pi wöchentlich neu zu verkabeln, bzw. den Ausgangszustand des EDV- Raums wieder herzustellen. Dadurch geht Unterrichtszeit verloren und außerdem können durch das häufige Ein- und Ausstecken Schäden an den Schul-PCs als auch an den Raspberry Pi entstehen. Ein weiterer Nachteil liegt darin, dass in dieser Konfiguration alle Aufgaben vom Raspberry Pi übernommen werden müssen – so zum Beispiel auch Recherchen während der Arbeit, die in einem Internetbrowser durchgeführt werden. Das führt dazu, dass die CPU ausgelastet ist und die Schüler häufig auf das System warten müssen.

### 2.3.2 Raspberry Pi im WLAN, Zugriff über XTerminal

Das ist jene Konfiguration welche für diese Arbeit gewählt wurde. In diesem Szenario wird der Raspberry Pi nur mit einem USB Wi-Fi-Adapter und dem Netzteil verbunden.

Gerät	Preis in €
Raspberry Pi (Modell B2)	32,88
SD-Karte	12,21
USB WLAN Adapter	10,70
Micro-USB Kabel	3,39
Netzteil	13,10
<b>Gesamtkosten/Arbeitsplatz</b>	<b>72,28</b>

Tabelle 6: Kosten bei Zugriff über XTerminal (WLAN)

Der Raspberry Pi wird gestartet und verbindet sich mit dem WLAN Router und somit mit dem Netzwerk. Die Schüler sitzen an ihren Windows-PCs und greifen auf den Linux Desktop über ein XTerminal-Programm zu (z.B. MobaXTerm, vgl. Kapitel 3.12).

Diese Konfiguration hat den Vorteil, dass keine Manipulationen an der Hardware des EDV-Raums notwendig sind. Die Schüler verbinden „ihren“ Raspberry Pi mit einem Pidget, starten ihn und können mit der Arbeit beginnen. Sollten während der Arbeit Recherchen notwendig sein (z.B. Nachschlagen der Syntax einer Schleife in Python), so können diese rasch am Windows-PC erledigt werden und die Ergebnisse (z.B. Codefragmente) mittels Kopieren/Einfügen in die Entwicklungsumgebung eingebracht werden. Ein weiterer Vorteil ist, dass in diesem Szenario der Raspberry Pi „mobil“ wird und sich nicht physisch vor dem Benutzer befinden muss. Der Raspberry Pi kann zum Beispiel auch in einem Roboter oder Fahrzeug eingebaut sein – er muss nur über das WLAN erreicht werden können.

Die Nachteile dieser Konfiguration sind, dass zuerst Software auf den Rechnern des EDV-Raums installiert werden muss (wobei das o.a. Programm MobaXTerm auch als Portable App zur Verfügung steht) und die Schüler zusätzlich den Umgang mit dieser Software erlernen müssen. Der Zugriff über ein XTerminal Programm und damit die „indirekte Arbeit“ am Raspberry Pi macht es für die Schüler schwieriger, die Konfiguration zu überblicken und zu verstehen.

### 2.3.3 Raspberry Pi im LAN, Zugriff über XTerminal

In der letzten Konfiguration ist der Raspberry Pi über die Ethernetchnittstelle mit dem Netzwerk verbunden.

Gerät	Preis in €
Raspberry Pi (Modell B2)	32,88
SD-Karte	12,21
Netzwerkkabel	2,60
Micro-USB Kabel	3,39
Netzteil	13,10
<b>Gesamtkosten/Arbeitsplatz</b>	<b>64,18</b>

Tabelle 7: Kosten bei Zugriff über XTerminal (LAN)

Auch hier verbindet sich der Raspberry Pi nach dem Systemstart mit dem Netzwerk und die Schüler greifen auf den Linux Desktop über ein XTerminal-Programm zu. Dazu ist es notwendig, dass zu allen Arbeitsplätzen eine zweite Netzwerkleitung für den Raspberry Pi installiert wird. Die Vor- und Nachteile dieser Konfiguration sind die gleichen wie beim vorherigen Szenario mit der Ausnahme, dass der Raspberry Pi hier nicht auf sich bewegende Objekte gesetzt werden kann.

### 3 Software Installation

In diesem Kapitel wird die Softwareinstallation des Raspberry Pi beschrieben. Zuerst wird eine Übersicht über die existierenden Betriebssysteme für den Raspberry Pi gegeben. Anschließend folgt eine Schritt-für-Schritt Anleitung für das Einrichten eines Betriebssystem-Images auf der SD-Karte, dem Durchführen von Grund- und Netzwerkeinstellungen am Raspberry Pi, der Installation von Zusatzsoftware, dem Anlegen weiterer Benutzer und dem Sichern der gesamten Installation.

Im letzten Abschnitt dieses Kapitels werden Hilfsprogramme vorgestellt, die einerseits für die Wartung und das Duplizieren der Installation benötigt werden und andererseits im Unterricht dazu verwendet werden können, um über das Netzwerk auf den Raspberry Pi zugreifen zu können.

Alle diese erwähnten Schritte sind – so sie auf unterschiedliche Art und Weise durchgeführt werden müssen – für die Betriebssysteme Windows und Mac OS X beschrieben.

#### 3.1 Übersicht Betriebssysteme

Es existieren mehrere Betriebssysteme für den Raspberry Pi. Die Raspberry Pi Foundation bietet im Downloadbereich ihrer Homepage [9] die folgenden vier Betriebssystemimages an:

- **Raspbian “wheezy”**  
Eine für den Raspberry Pi optimierte Debian Distribution, welche die grafische Benutzeroberfläche Lightweight X11 Desktop Environment (LXDE), den Webbrowser Midori und Softwareentwicklungswerkzeuge (Scratch, Idle) enthält.  
Dieses Betriebssystemimage wird für Einsteiger empfohlen. Es ist auch die Basis für die im nächsten Abschnitt beschriebene Installation.
- **Soft-float Debian “wheezy”**  
Dieses Image ist identisch zum Raspbian "wheezy" Image, allerdings verwendet es ein soft-float Application Binary Interface (ABI). Dieses muss dann eingesetzt werden, wenn Software verwendet wird (z.B. Oracle JVM), die hard-float ABI nicht unterstützt.
- **Arch Linux ARM**  
Dieses Image basiert auf Arch Linux, welches Einfachheit (der Bootvorgang vom Einschalten bis zur Eingabeaufforderung dauert ungefähr 10 Sekunden) und volle Kontrolle des Benutzers zum Ziel hat. Es ist daher für Einsteiger eher nicht geeignet.
- **RISC OS**  
Eine für Benutzer des Raspberry Pi freie Version des RISC Betriebssystems.

Neben den Images, die der Hersteller des Raspberry Pi zur Verfügung stellt, sollen hier noch zwei interessante Versionen erwähnt werden:

- Raspbmc [10]: dieses Betriebssystem verwandelt den Raspberry Pi in ein Mediacenter.
- Razdroid [11]: ist der Versuch, das Betriebssystem Android für die Hardware des Raspberry Pi zu portieren.

Ein Vergleich der oben genannten und noch weiterer Betriebssysteme für den Raspberry Pi findet sich auf den Seiten von Embedded Linux. [12]

### 3.2 Betriebssystem installieren – Windows

Wie bereits erwähnt wird im Rahmen dieser Arbeit Raspian „wheezy“ verwendet. Um dieses Betriebssystem auf dem Raspberry Pi zu installieren wird zuerst das Image auf einen Windows-PC heruntergeladen und entpackt. Anschließend muss das Image mit einem Imagetool auf die SD-Karte geschrieben werden, z.B. mit „USB Image Tool“ (vgl. Abbildung 8).

#### Hinweise:

- Das Image darf **nicht** via „drag and drop“ auf die SD-Karte kopiert werden. Nur die weiter unten beschriebene Vorgehensweise liefert eine bootfähige SD-Karte.
- Durch das Beschreiben der SD-Karte mit dem Betriebssystemimage werden alle Daten auf der SD-Karte überschrieben.
- Windows-Rechner zeigen bei der fertig beschriebenen SD-Karte eine Größe von nur 55 bis 75 MB an. Dieser Umstand rührt daher, dass der größte Teil der Karte in einem Dateisystem formatiert wurde, das für Windows-Rechner nicht lesbar bzw. sichtbar ist.
- Um den gesamten Speicherplatz der SD-Karte wieder für Windows nutzen zu können, muss die Karte (ebenfalls mittels eines Imagetools) zurückgesetzt und anschließend formatiert werden.

Um das Image auf die SD-Karte zu schreiben, wird die zu beschreibende SD-Karte in der linken Spalte ausgewählt und anschließend „Restore“ geklickt. Im folgenden Dialog wird das zu schreibende Image ausgewählt und die Rückfrage bestätigt, dass die SD-Karte überschrieben werden darf.

Nachdem das Betriebssystemimage auf die Karte geschrieben wurde, wird das Image Tool beendet, die Karte ausgeworfen und in den Raspberry Pi eingesetzt. Anschließend wird der Raspberry Pi mit Tastatur, Maus, Monitor und Netzwerk verbunden und – zum Schluss – durch Verbinden mit dem Netzteil in Betrieb genommen.

### 3.3 Betriebssystem installieren – Mac OS X

Auch hier muss zuerst das Betriebssystemimage heruntergeladen und entpackt werden. Nach dem Herunterladen des Images muss vor dem Anschließen des SD-Kartenlesers noch ein Terminalfenster geöffnet werden.

Mit dem Befehl

```
cd Downloads/
```

wird in das Verzeichnis gewechselt, in dem das Betriebssystemimage gespeichert ist.

```
diskutil list
```

liefert eine Auflistung der Datenträger des Systems. Diese Auflistung kann beispielsweise so aussehen:

```
Last login: Fri May 10 15:56:10 on ttys000
Klauss-Mac-mini:~ MacKlaus$ diskutil list
/dev/disk0
```

#:	TYPE	NAME	SIZE	IDENTIFIER
0:	GUID_partition_scheme		*1.0 TB	disk0
1:	EFI		209.7 MB	disk0s1
2:	Apple_HFS	Macintosh HD	699.3 GB	disk0s2
3:	Apple_Boot	Recovery HD	650.0 MB	disk0s3
4:	Microsoft Basic Data	BOOTCAMP	300.0 GB	disk0s4

Listing 1: Auflistung der Datenträger des Systems

Im nächsten Schritt wird der Kartenleser mit dem Rechner verbunden. Sobald das Kartensymbol am Schreibtisch erscheint, wird der Befehl `diskutil list` erneut aufgerufen.

```
Klauss-Mac-mini:~ MacKlaus$ diskutil list
/dev/disk0
```

#:	TYPE	NAME	SIZE	IDENTIFIER
0:	GUID_partition_scheme		*1.0 TB	disk0
1:	EFI		209.7 MB	disk0s1
2:	Apple_HFS	Macintosh HD	699.3 GB	disk0s2
3:	Apple_Boot	Recovery HD	650.0 MB	disk0s3
4:	Microsoft Basic Data	BOOTCAMP	300.0 GB	disk0s4

```
/dev/disk1
```

#:	TYPE	NAME	SIZE	IDENTIFIER
0:	<b>FDisk_partition_scheme</b>		<b>*15.5 GB</b>	<b>disk1</b>
1:	Windows_FAT_32		58.7 MB	disk1s1
2:	Linux		1.9 GB	disk1s2

Listing 2: Auflistung der Datenträger des Systems (inkl. SD-Karte)

Durch dieses Prozedare ist ersichtlich, dass die zu beschreibende Karte `/dev/disk1` mit knapp 16GB ist. Das ist wichtig, weil sonst irrtümlich ein falsches Laufwerk überschrieben werden könnte!

Als nächstes müssen alle Partitionen der SD-Karte aus dem Dateisystem entfernt werden. Für obiges Beispiel sind dazu die folgenden beiden Befehle nötig:

```
sudo umount /dev/disk1s1
sudo umount /dev/disk1s2
```

Als letzter Schritt wird das Raspian „wheezy“ Image auf die SD-Karte geschrieben. Wiederum für das obige Beispiel lautet der Befehl:

```
sudo dd bs=1m if=2013-02-09-wheezy-raspbian.img of=/dev/disk1
```

wobei hier `bs` die Blocksize der Übertragung angibt, `if` ist das Inputfile, `of` das Outputfile. Wichtig ist, dass für den Parameter Inputfile das entpackte Betriebssystemimage (Dateiendung `*.img`) und nicht die heruntergeladene Zip-Datei (`*.zip`) angegeben wird. Das Beschreiben der SD-Karte dauert einige Minuten.

Analog wie bei der Installation unter Windows, muss auch hier nach dem Beschreiben der SD-Karte das Terminalprogramm beendet, die Karte ausgeworfen und in den Raspberry Pi eingesetzt werden. Anschließend wird der Raspberry Pi mit Tastatur, Maus, Monitor und Netzwerk verbunden und – zum Schluss – durch Verbinden mit dem Netzteil in Betrieb genommen.

### 3.4 Grundeinstellungen

Beim ersten Start des Raspberry Pi erscheint das folgende Konfigurationsprogramm in dem die Grundeinstellungen des Systems vorgenommen werden können (vgl. Abbildung 2).

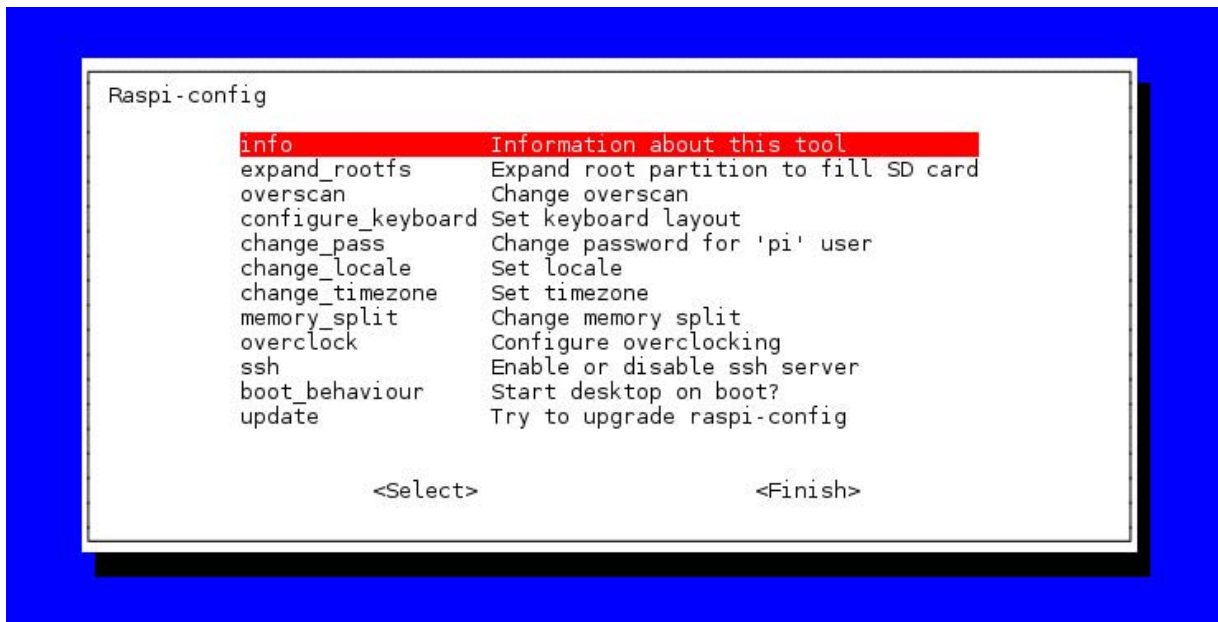


Abbildung 2: Raspi-config Programm

- **update**  
versucht, die neueste Version des Konfigurationsprogramms zu installieren. Falls der Raspberry Pi schon Verbindung zum Internet hat, sollte dieser Punkt als erstes durchgeführt werden.
- **info**  
Information über das Konfigurationstool
- **expand\_rootfs**  
erweitert die Größe der Systempartition (2GB) auf die Größe der verwendeten SD-Karte. Nach dem Aufruf dieses Punktes wird die Erweiterung vorbereitet. Die tatsächliche Erweiterung wird erst beim nächsten Systemstart durchgeführt.
- **overscan**  
in diesem Punkt kann overscan ein- bzw. ausgeschaltet werden. Overscan produziert einen schwarzen Rahmen um die Bildausgabe. Dieser Rahmen war bei analogen TV-Geräten und Röhrenmonitoren notwendig, um einen geraden Bildabschluss zu erreichen. Bei TFT-Monitoren kann overscan deaktiviert werden. Die Änderung erfolgt beim nächsten Systemstart.
- **configure\_keyboard**  
legt Tastaturmodell und Sprache der Tastatur (z.B. Logitech Cordless Desktop, German (Austria)) und die Funktion einiger Tasten und Tastenkombinationen (AltGr, Compose Key, Strg + Alt + Backspace) fest.
- **change\_pass**  
Der Benutzer „pi“ mit Passwort „raspberry“ ist bereits angelegt. Mit diesem Punkt kann das Passwort geändert werden.



- **change\_locale**  
bestimmt Zeichencodierung und Betriebssystemsprache. Für ein „österreichisches“ System wird der Eintrag „de\_AT.UTF8 UTF8“ mit der Leertaste markiert. Die Markierung bei "en\_GB.UTF-8 UTF-8" kann - ebenfalls mit der Leertaste - entfernt werden. Die Auswahl wird mit der Returnntaste abgeschlossen.
- **change\_timezone**  
Zeitzone auf Europa / Wien einstellen.
- **memory\_split**  
der Hauptspeicher des Raspberry Pi wird gemeinsam von Prozessor und Grafikprozessor (GPU) genutzt. In diesem Punkt wird festgelegt, welcher Anteil des Hauptspeichers der GPU zugewiesen wird. Eine Änderung wird beim nächsten Neustart wirksam.
- **overclock**  
hier kann der Prozessortakt zwischen 700MHz (Standardwert) und 1GHz eingestellt werden. Die Raspberry Pi Foundation warnt davor, dass Übertakten zu Verkürzung der Lebenszeit des Raspberry Pi und zu Systeminstabilitäten führen kann. Für die Aufgaben in dieser Arbeit ist es nicht notwendig, den Prozessor zu übertakten.
- **ssh**  
die secure shell (SSH) bietet die Möglichkeit, über das Netzwerk auf die Textkonsole des Systems zuzugreifen. Die Verbindung wird automatisch verschlüsselt. Um diesen Zugriff auf den Raspberry Pi zu ermöglichen, muss der SSH-Serverdienst laufen. Dieser kann mit diesem Punkt aktiviert werden.
- **boot\_behaviour**  
legt fest, ob beim Systemstart die grafische Benutzeroberfläche LXDE (Lightweight X11 Desktop Environment) automatisch gestartet werden soll.

Zum Beenden des Konfigurationsprogrammes muss die Tabulatortaste so oft gedrückt werden, bis „Finish“ rot hinterlegt ist. Anschließend dann das Programm durch Drücken der Returnntaste beendet werden. Um die Änderungen der Einstellungen wirksam zu machen, muss das System neu gestartet werden.

#### **Hinweise:**

- Das Konfigurationsprogramm kann auch zu einem späteren Zeitpunkt über folgenden Konsolenbefehl gestartet werden:  
`sudo raspi-config`
- Die grafische Benutzeroberfläche LXDE kann manuell über den Konsolenbefehl gestartet werden:  
`startx`
- `sudo` steht für `substitute user do` und ist ein Linux Konsolenbefehl um Prozesse mit den Rechten des Superusers `root` zu starten ohne dessen Passwort zu benötigen.

### **3.5 Netzwerkeinstellungen**

Wenn der Raspberry Pi über ein Ethernetkabel mit einem Router verbunden ist und auf diesem ein DHCP Server läuft, dann verbindet sich der Raspberry Pi automatisch mit dem Netz-

werk, da der Pi automatisch eine freie IP- Adresse zugewiesen bekommt. In den Arbeitsszenarien, die in Kapitel 2.3 beschrieben wurden ist es jedoch zweckmäßiger, jedem der verwendeten Pi-Geräte eine statische IP-Adresse zuzuweisen. Dazu wird ein LX-Terminalfenster geöffnet (das Programm LX-Terminal befindet sich am Schreibtisch des LXDE) und die Konfigurationsdatei für die Netzwerk-Interfaces mit dem Texteditor *nano* bearbeitet:

```
sudo nano /etc/network/interfaces
```

In der Datei „interfaces“ muss die Zeile

```
iface eth0 inet dhcp
```

beispielsweise auf folgende Einstellungen abgeändert werden:

```
iface eth0 inet static
    address 192.168.1.205
    netmask 255.255.255.0
    gateway 192.168.1.1
    dns-nameservers 192.168.1.1
```

Abschließend wird die Datei mit Strg + O unter gleichem Namen gespeichert, der Editor *nano* mit Strg + X beendet und der Raspberry Pi mit

```
pi@raspberrypi ~ $ sudo reboot
```

neu gestartet.

Um eine WLAN Verbindung herzustellen, muss vor dem Einschalten des Raspberry Pi ein WLAN Adapter an einen der USB-Ports eingesteckt werden. Am Schreibtisch des LXDE befindet sich das Programm *Wi-Fi Config* mit dem sich die WLAN Verbindung einrichten lässt.

### Hinweise:

- Wenn ein Ethernetkabel eingesteckt ist, so wird aus Stromspargründen der WLAN Adapter deaktiviert und eine WLAN Verbindung ist nicht mehr möglich.
- Wird der WLAN Adapter im Betrieb eingesteckt, so stürzt der Raspberry Pi ab und startet neu.

## 3.6 Paketverwaltung installieren

Die Software eines Linux-Systems – also auch die des Raspberry Pi – ist in Paketen organisiert. Eine Paketverwaltung ermöglicht es, Software zu installieren, zu aktualisieren oder zu entfernen. In diesem Schritt wird „Synaptic“, eine Paketverwaltung mit grafischer Benutzeroberfläche, installiert.

Dazu wird in einem LX-Terminalfenster der Befehl

```
sudo apt-get install synaptic
```

eingegeben und im folgenden Installationsprozess werden alle Eingabeaufforderungen mit „j“ beantwortet. Nach erfolgter Installation kann die Paketverwaltung über den Menüeintrag „Desktopmenü > Einstellungen > Synaptic Paketverwaltung“ gestartet werden.

### 3.7 Software aktualisieren

Die Paketverwaltung Synaptic (vgl. Abbildung 3) bietet die Möglichkeit, die installierten Pakete (samt Betriebssystem) auf Updates zu prüfen und ggf. zu aktualisieren.

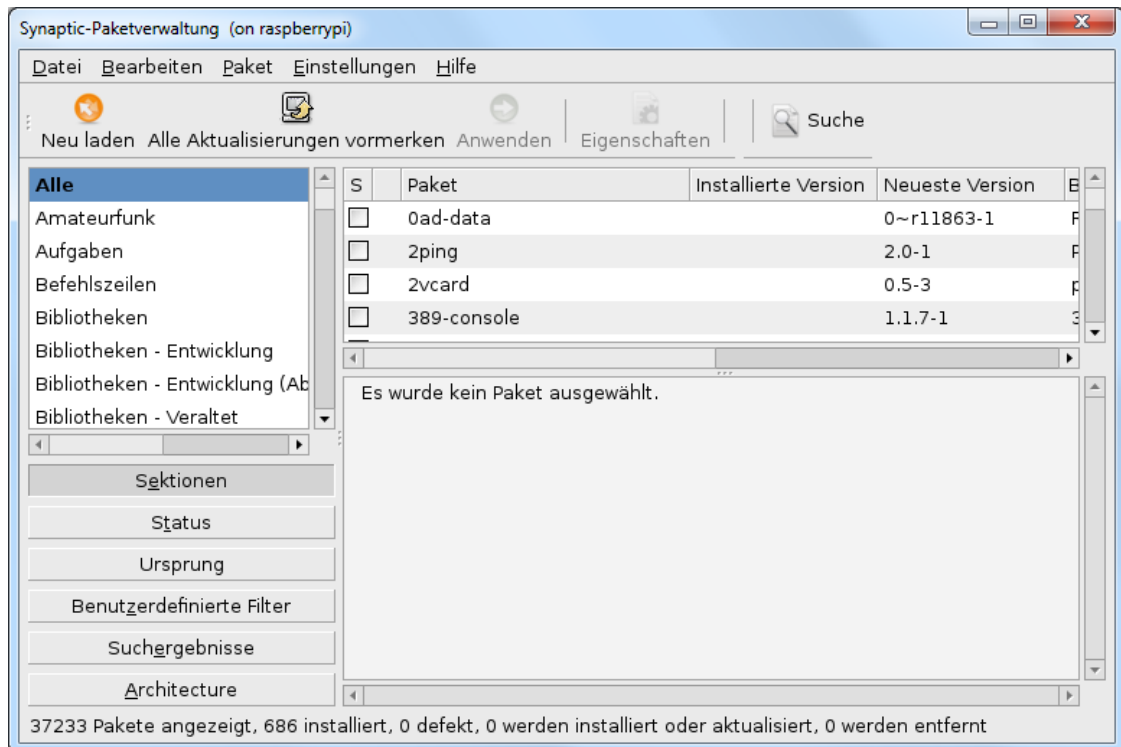


Abbildung 3: Synaptic Paketverwaltung

Die Aktualisierung läuft in drei Schritten ab. Zuerst wird nach dem Programmstart die Schaltfläche „Neu laden“ angewählt. In diesem Schritt werden zu den installierten Paketen die aktualisierten Informationen heruntergeladen und verglichen.

Im zweiten Schritt wird durch den Befehl „Alle Aktualisierungen vormerken“ ein Fenster geöffnet, in dem die zu aktualisierenden Pakete angezeigt und zur Aktualisierung vorgemerkt werden können.

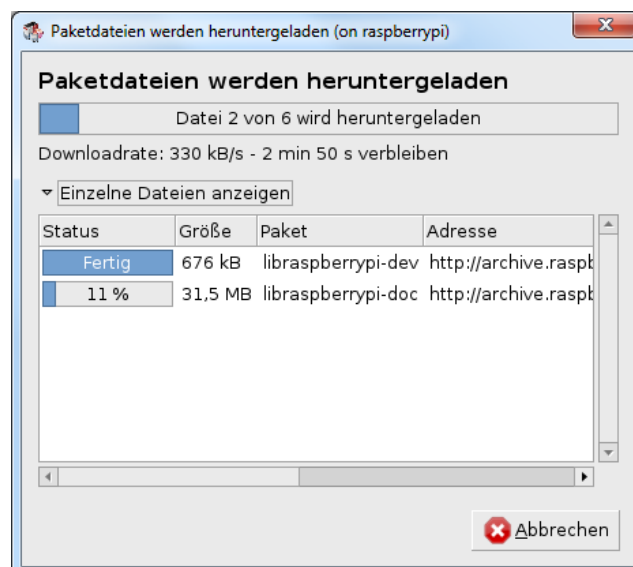


Abbildung 4: Pakete herunterladen

Nach dem Vormerken werden schließlich im dritten Schritt durch Klicken auf die Schaltfläche „Anwenden“ die aktualisierten Pakete heruntergeladen und installiert (vgl. Abbildung 4).

#### Hinweis:

- Eine Softwareaktualisierung ist auch mittels folgender Konsolenbefehle möglich
 

```
sudo apt-get update      # Neueinlesen der Paketlisten
sudo apt-get upgrade    # Installierte Pakete aktualisieren
sudo apt-get autoremove # Deinstallation ungenutzter Abhängigkeiten
```

### 3.8 Zusatzpakete installieren

Für das Lösen der Programmieraufgaben dieser Arbeit benötigen die Schüler noch eine integrierte Entwicklungsumgebung (IDE).

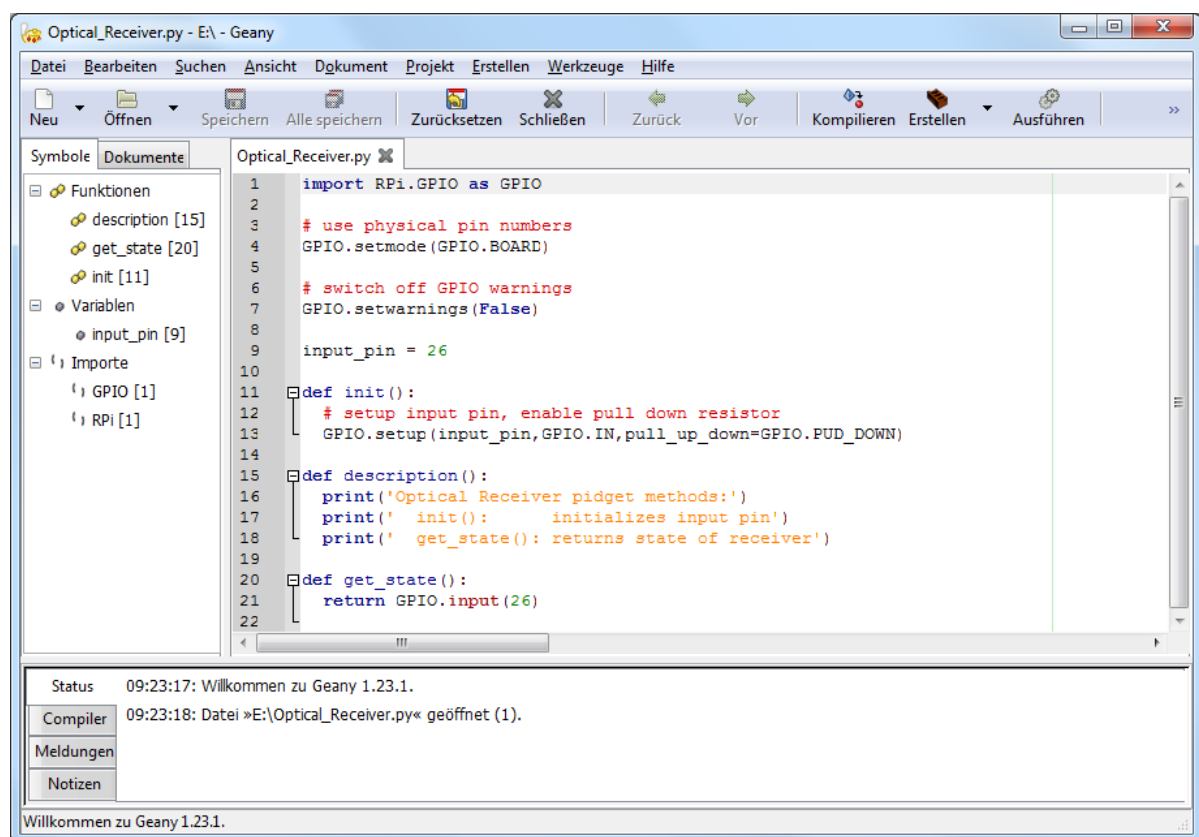


Abbildung 5: IDE Geany

Die IDE *Geany* (vgl. Abbildung 5) ist eine schlanke und übersichtliche Entwicklungsumgebung, die für die Schüler wesentliche Hilfestellungen bietet: Syntax Highlighting, Code-Vervollständigung, Autovervollständigung von Verzweigungen und Schleifen etc. und zusätzlich ist für *Geany* ein Debugging-PlugIn verfügbar.

Um *Geany* zu installieren, wird *Synaptic* gestartet und als Suchbegriff *Geany* eingegeben. Aus dem Suchergebnis werden nun folgende Pakete durch Rechtsklick für die Installation vorge-merkt:

- *geany*
- *geany-common*
- *geany-plugin-debugger*

Mit dem Befehl „Anwenden“ wird die Installation durchgeführt.

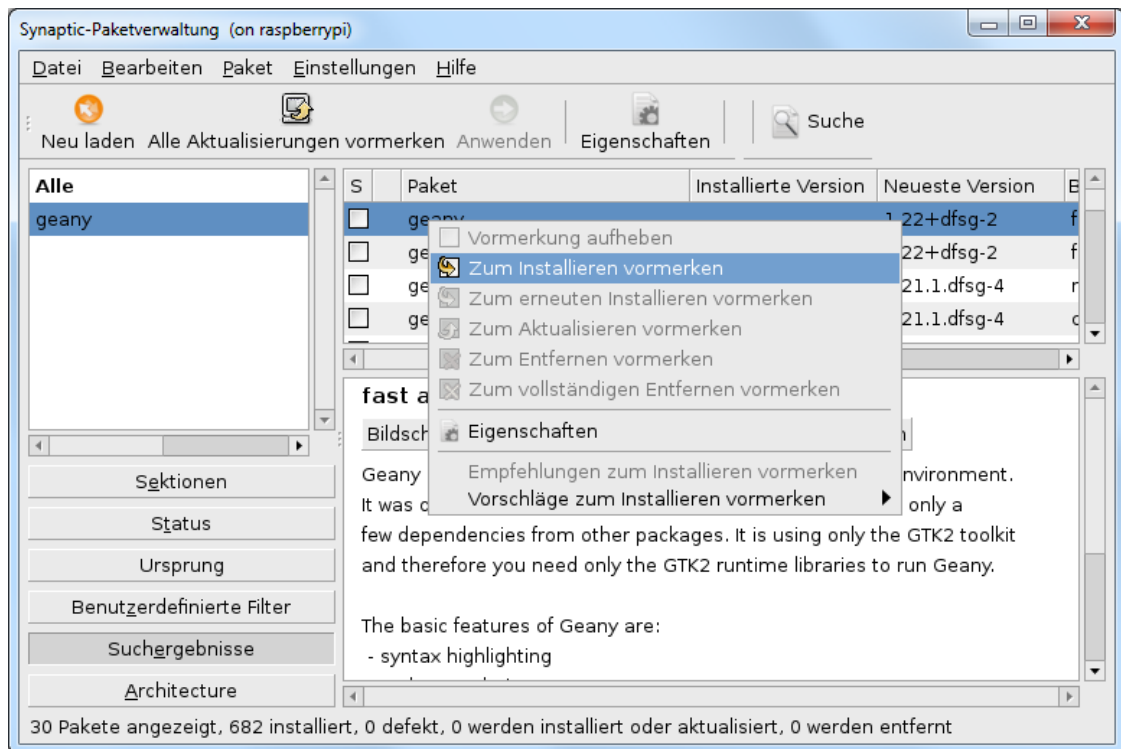


Abbildung 6: Pakete vormerken

### Hinweise:

- Mit dem Vormerken des Pakets *geany* wird auch automatisch *geany-common* vorge-merkt.
- Um den GPIO-Port des Raspberry Pi ansprechen zu können, war es bei früheren Distri-butionen erforderlich, die Pakete *python-rpi.gpio* und *python3-rpi.gpio* zu installie-ren. Die Vorgangsweise ist analog wie bei der Installation von *Geany*.
- Für den Raspberry Pi gibt es noch viele interessanter Softwarepakete, die über Synap-tic installiert und im Unterricht eingesetzt werden könnten (z.B. *LibreOffice*, *Gimp*, *Geogebra*, *education-physics*, *education-development*, *childsplay*, ...).
- Eine weitere Quelle für Software ist der *Pi Store*. Die dazugehörige Anwendung be-findet sich am Desktop des LXDE.

### 3.9 IDE Geany mit root-Privilegien ausführen

Um den GPIO Port des Raspberry Pi aus Python ansprechen zu können, muss das Pythonpro-gramm mit root-Rechten ausgeführt werden. Dies kann erreicht werden, indem das Programm mit folgendem Befehl aus der Konsole gestartet wird:

```
sudo python path/filename.py
```

Eine weitere Möglichkeit, die den Schüler das Arbeiten erleichtert, ist es der IDE *Geany* für die Ausführung die entsprechenden Rechte zu geben. Dazu wird mit dem Texteditor *nano* die folgende Datei geöffnet

```
sudo nano /usr/share/applications/geany.desktop
```

und der Eintrag

```
Exec=geany %F
```

auf

```
Exec=gksudo geany %F
```

abgeändert, die Datei mit Strg + O unter gleichem Namen gespeichert und der Editor *nano* mit Strg + X beendet. *gksudo* ist wie *sudo*, nur für Programme mit grafischer Benutzerschnittstelle.

### 3.10 Pythoncompiler ändern

Zum Zeitpunkt des Verfassens dieser Arbeit existieren zwei Python-Versionen – Version 2.7 (die letzte 2.x Version) und Version 3.3 (die aktuelle 3.x Version). Die Version 3.x ist nicht vollständig rückwärtskompatibel zu 2.x.

Die IDE *Geany* verwendet standardmäßig Python 2.x. Um den verwendeten Compiler auf Version 3.x umzustellen, sind folgende Schritte notwendig: zuerst wird in einem Terminalfenster das folgende Kommando eingegeben:

```
pi@raspberrypi ~ $ sudo nano /usr/share/geany/filetypes.python
```

Am Ende dieser Datei die beiden Zeilen mit „#“ auskommentieren

```
# compiler=python -m py_compile "%f"
# run_cmd=python "%f"
```

und folgende Zeilen einfügen:

```
compiler=python3 -c "import py_compile; py_compile.compile('%f')"
run_cmd=python3 "%f"
```

Die Datei mit Strg + O unter gleichem Namen speichern und den Editor *nano* mit Strg + X beenden. Damit ist die Softwareinstallation des Raspberry Pi abgeschlossen.

#### Hinweise:

- Alle Musterlösungen dieser Arbeit sind in Python 3 kompatibler Software geschrieben.
- Der Konsolenaufruf  

```
pi@raspberrypi ~ $ sudo python path/filename.py
```

verwendet den Python 2.7 Compiler. Um den Compiler für Python 3.x zu verwenden, muss der Befehl auf  

```
pi@raspberrypi ~ $ sudo python3 path/filename.py
```

geändert werden.

### 3.11 Benutzer hinzufügen

Um zusätzliche Benutzerkonten anzulegen ist der folgende Konsolenbefehl einzugeben

```
pi@raspberrypi ~ $ sudo adduser benutzername
```

Abbildung 7 zeigt das Anlegen des neuen Benutzers *student1*

```

pi@raspberrypi: ~ (on raspberrypi)
Datei Bearbeiten Reiter Hilfe
pi@raspberrypi ~ $ sudo adduser student1
Lege Benutzer »student1« an ...
Lege neue Gruppe »student1« (1003) an ...
Lege neuen Benutzer »student1« (1002) mit Gruppe »student1« an ...
Erstelle Home-Verzeichnis »/home/student1« ...
Kopiere Dateien aus »/etc/skel« ...
Geben Sie ein neues UNIX-Passwort ein:
Geben Sie das neue UNIX-Passwort erneut ein:
passwd: Passwort erfolgreich geändert
Benutzerinformationen für student1 werden geändert.
Geben Sie einen neuen Wert an oder drücken Sie ENTER für den Standardwert
Vollständiger Name []:
Zimmernummer []:
Telefon geschäftlich []:
Telefon privat []:
Sonstiges []:
Sind die Informationen korrekt? [J/n] J
pi@raspberrypi ~ $

```

Abbildung 7: Benutzer hinzufügen

Mit diesem Prozedere ist der Benutzer `student1` angelegt worden und hat auch sein eigenes Verzeichnis unter `/home/student1` erhalten. Was noch fehlt, ist dem Benutzer die Berechtigung zu geben, Programme mit root-Privilegien auszuführen. Dazu muss die folgende Datei geöffnet

```
pi@raspberrypi ~ $ sudo visudo
```

der Eintrag

```
#includedir /etc/sudoers.d
pi ALL=(ALL) NOPASSWD: ALL
```

um die Zeile

```
student1 ALL=(ALL) NOPASSWD: ALL
```

ergänzt und die Datei unter gleichem Namen abgespeichert werden. Diese Art der Rechtevergabe dient der Einfachheit und ist nicht unbedingt „best practice“. Der Fokus dieser Arbeit ist jedoch nicht größtmögliche Sicherheit, sondern die Nutzung des Pi und der Pidgets.

Mit dem Anlegen weiterer Benutzer ist der Raspberry Pi fertig installiert und eingerichtet. Benutzer von Windows-Rechnern können die sich auf der SD-Karte befindliche Installation nun mittels USB Image Tool sichern und für alle im Unterricht verwendeten Raspberry Pi duplizieren. Bei Rechnern unter Mac OS X ist das Sichern der gesamten Installation über folgenden Konsolenbefehl möglich (SD-Karte als `/dev/disk1` im Dateisystem):

```
sudo dd bs=1m if=/dev/disk1 of=Speicherpfad/Dateiname.img
```

#### Hinweis:

- Bedingt durch das Duplizieren der Installation ist zu beachten, dass jedem Raspberry Pi beim ersten Einschalten eine eigene IP Adresse zugewiesen werden muss.

### 3.12 Hilfsprogramme

Im letzten Abschnitt dieses Kapitels werden noch einige Hilfsprogramme vorgestellt, die für Wartung und Fernzugriffe auf den Raspberry Pi verwendet werden können. Alle diese Programme sind entweder frei erhältlich oder schon Bestandteil der jeweiligen Betriebssysteme.

Der Abschnitt ist in zwei Sektionen unterteilt: Hilfsprogramme für das Betriebssystem Windows und Hilfsprogramme für Mac OS X.

### 3.12.1 Hilfsprogramme Windows

#### USB Image Tool [13]

Wie schon erwähnt, kann mit diesem Programm das Betriebssystem für den Raspberry Pi auf die SD-Karte geschrieben werden. Zuerst wird in der linken Seite des Programmfensters das Speichermedium ausgewählt, auf das das Image geschrieben werden soll. Nach dem Ausführen des Befehls „Restore“ muss der Benutzer ein Betriebssystemimage auswählen und noch bestätigen, dass das Speichermedium überschrieben werden darf.

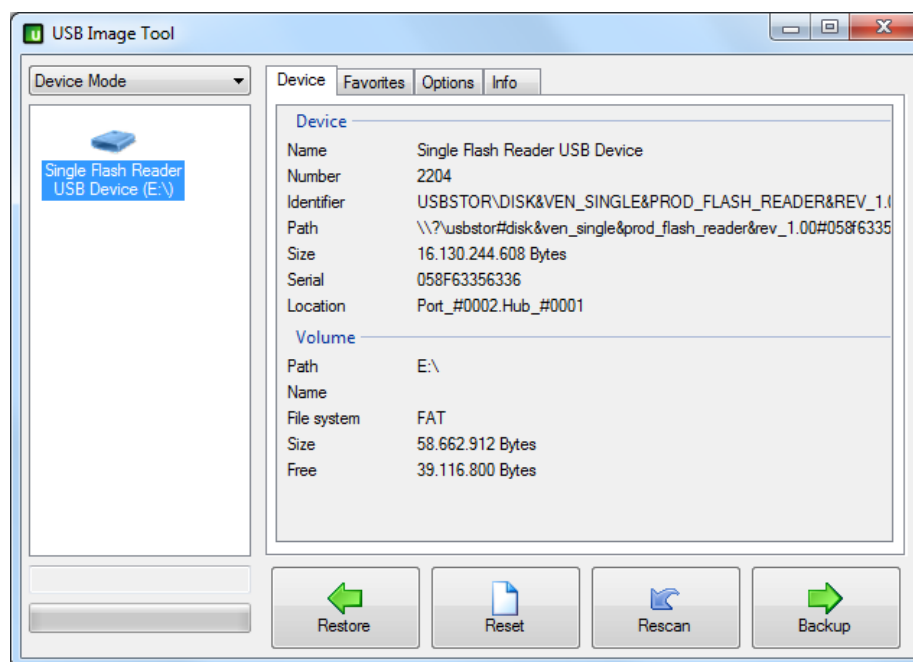


Abbildung 8: USB Image Tool

Die Funktionen der weiteren Schaltflächen:

- „Reset“ gibt den gesamten Speicherplatz der SD-Karte für die Nutzung unter Windows frei (löscht also die Linux Partition).
- „Rescan“ sucht nach neuen Geräten am USB.
- „Backup“ schreibt den Inhalt der SD-Karte als Image auf die Festplatte. Diese Funktion kann genutzt werden, um die fertige Installation zu sichern und in der Folge auf beliebig viele Raspberry Pi zu duplizieren.

#### PuTTY [14]

PuTTY ist ein Secure Shell-Programm für Microsoft Windows (vgl. Abbildung 9). Es dient dazu, eine Verbindung von einem Windows-Rechner zum Secure-Shell-Server des Raspberry Pi herzustellen. Vor dem Verbindungsaufbau wird die Identität des Benutzers überprüft.



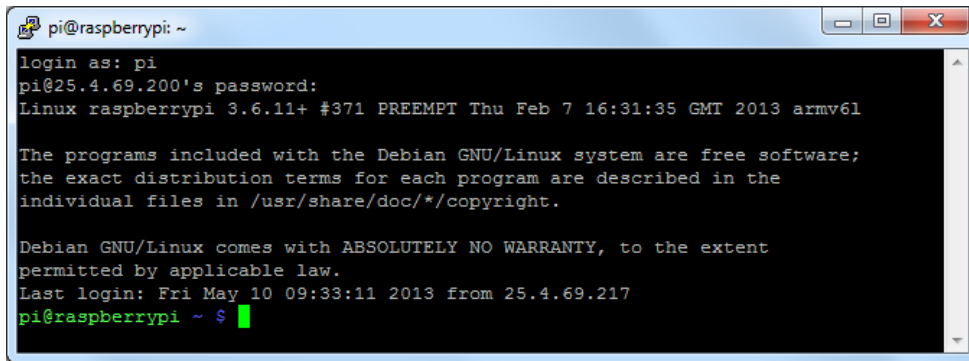


Abbildung 9: PuTTY

In der textorientierten Terminal-Sitzung können Befehle abgesetzt werden, die dann am Raspberry Pi ausgeführt werden. Für eine grafische Ausgabe ist ein X-Server notwendig, der auf dem Client-Rechner läuft (siehe MobaXterm).

Im Konfigurationsfenster können IP-Adresse des Zielrechners, Port und Verbindungstyp eingestellt und für die Wiederverwendung gespeichert werden (vgl. Abbildung 10).

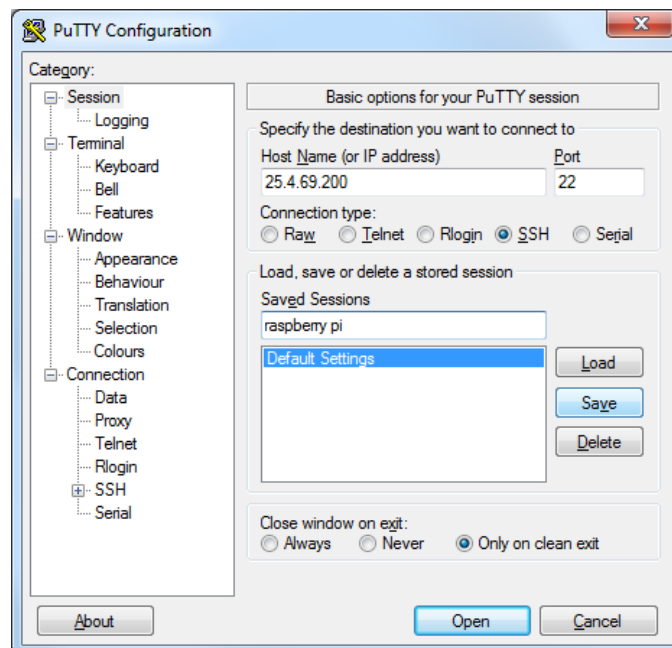


Abbildung 10: PuTTY Configuration

## MobaXterm [15]

MobaXterm ist ein portabler X-Server für Windows. Das X11-Forwarding läuft über OpenSSH. Darüber hinaus bietet MobaXterm wichtige Unix-Kommandos, ein tabbing-fähiges Terminal, einen Session-Manager, File Transfer Protocol (FTP), Secure File Transfer Protocol (SFTP) und das Remote Netzwerk-Tool VNC an (vgl. Abbildung 11).

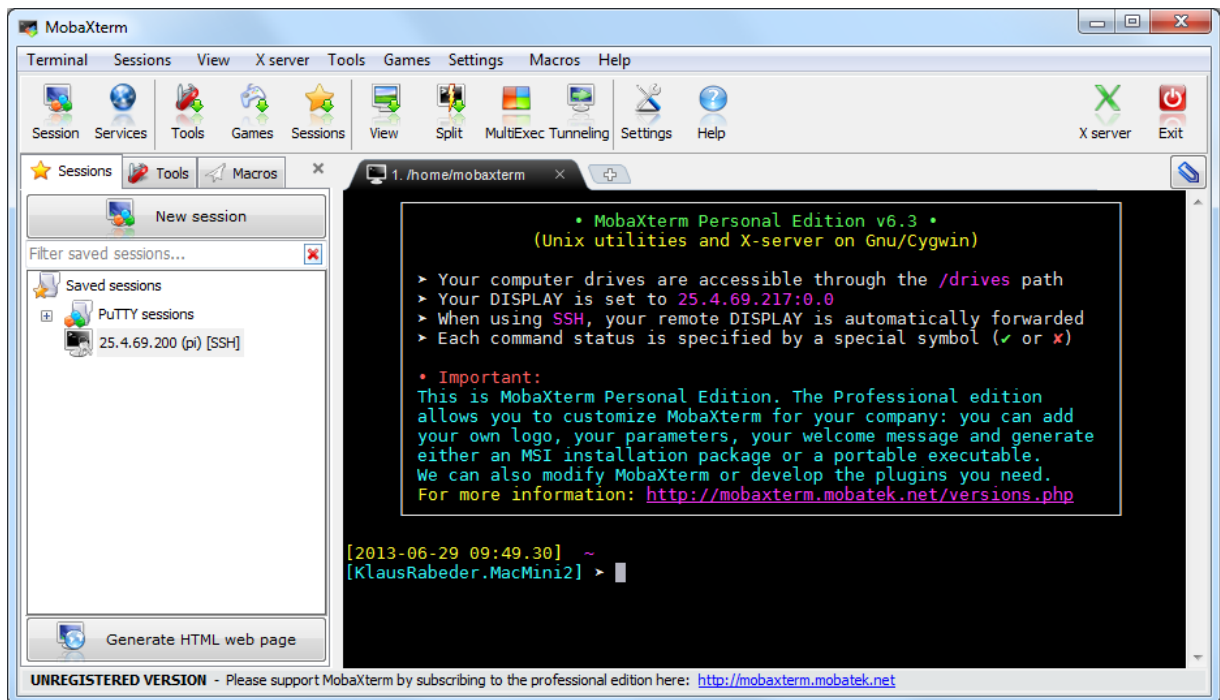


Abbildung 11: MobaXterm

Um eine XWindow-Session zu starten, muss der Befehl „New session“ ausgeführt werden, dann der Verbindungstyp SSH ausgewählt und die Host-Adresse und der Benutzername angegeben werden (vgl. Abbildung 12).

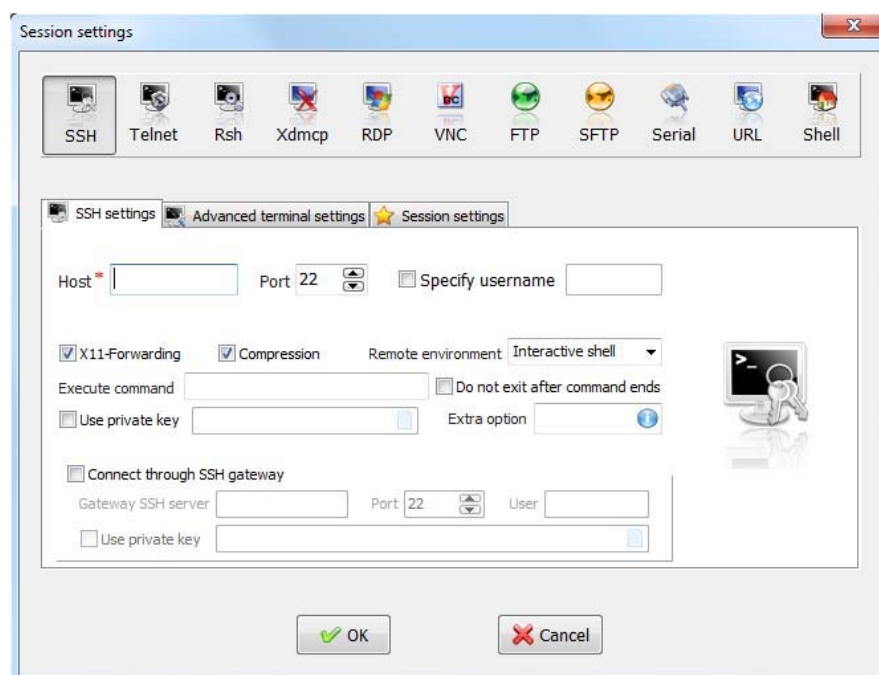


Abbildung 12: MobaXterm Session settings

Nach Bestätigung mit „OK“ muss sich der Benutzer am Raspberry Pi authentifizieren und kann dann durch Eingabe von

lxsession&

den Desktop des Raspberry Pi importieren.

## WinSCP [16]

Windows Secure CoPy (WinSCP) ist ein SSH, SFTP und FTP Client, der einen geschützten Datentransfer zwischen einem Windows PC und dem Raspberry Pi ermöglicht. Das Programm kann auch Ordner zwischen den beiden Systemen synchronisieren (skriptgesteuert) – diese Funktion kann beispielsweise das Verteilen von Aufgabenstellungen oder das „Einsammeln“ und Sichern der Ausarbeitungen der Schüler automatisch übernehmen. Die automatisierte Übertragung von Dateien ist auf der Herstellerseite beschrieben. [17]

Um eine Sitzung zu starten, müssen das Übertragungsprotokoll, die Adresse des Raspberry Pi, Port, Benutzername und Kennwort eingegeben werden (vgl. Abbildung 13). Diese Einstellungen lassen sich für die Wiederverwendung speichern.

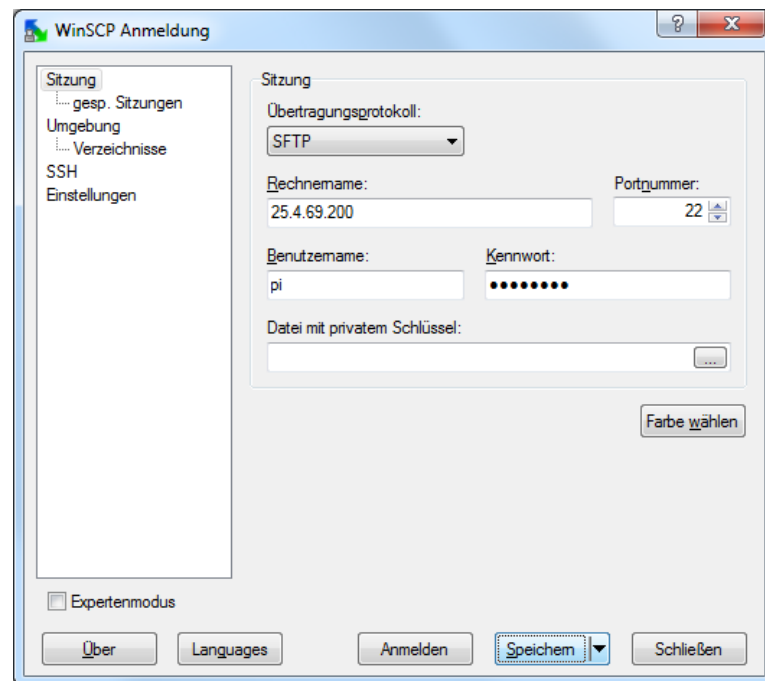


Abbildung 13: WinSCP Anmeldung

Für die Dateiübertragung stehen zwei Programmoberflächen zur Verfügung: eine im Stil des Windows-Explorers und eine im Stil des Norton-Commander, so wie sie in Abbildung 14 dargestellt ist. Dabei ist das Fenster vertikal geteilt – auf der linken Seite ist das Dateisystem des lokalen Rechners dargestellt, auf der rechten jenes des Raspberry Pi.

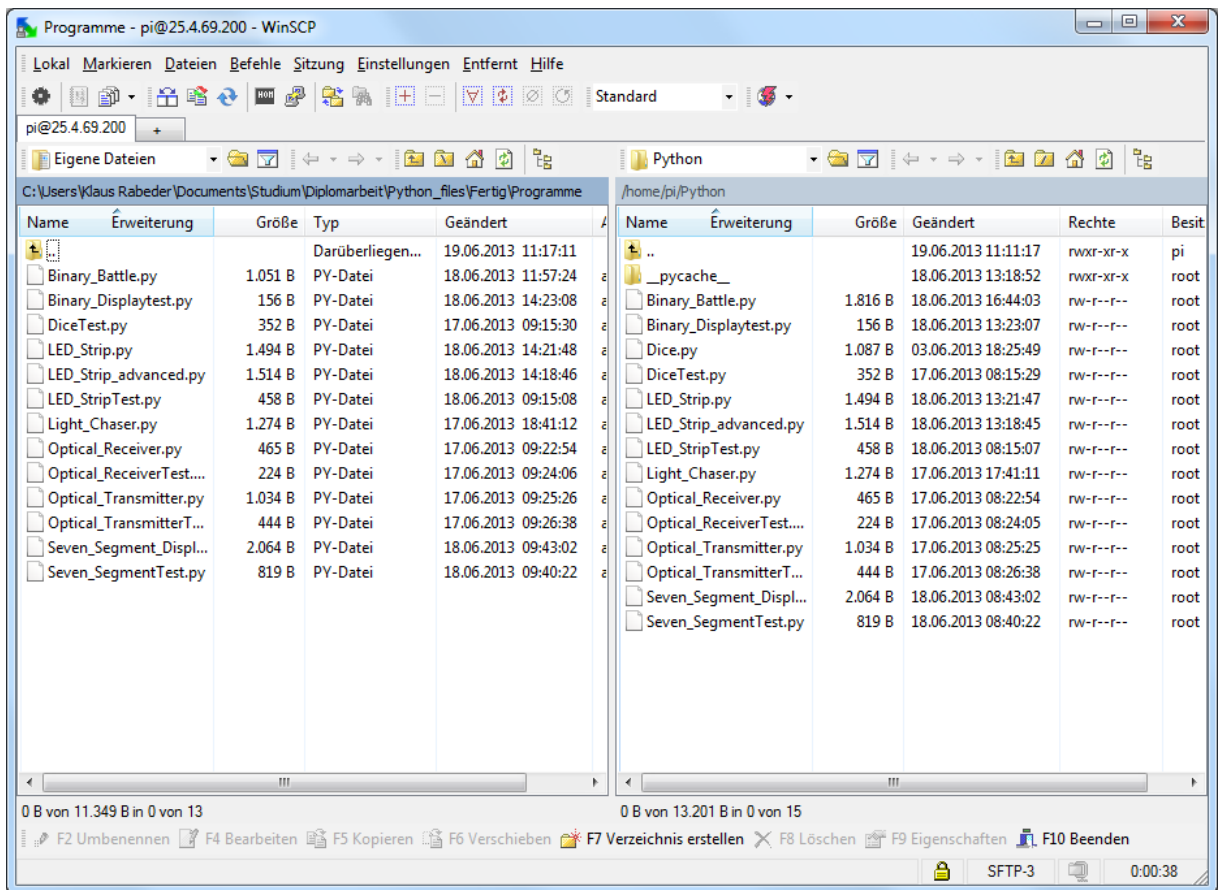


Abbildung 14: WinSCP

### 3.12.2 Hilfsprogramme für Mac OS X

#### RPi SD Card Builder [18]

Dieses Programm ist eine Alternative zur in Kapitel 3.3 beschriebenen Vorgangsweise über das Terminal. Der RPi SD Card Builder kann unter der angegebenen Adresse heruntergeladen werden, eine Schritt-für-Schritt Anleitung findet sich ebenfalls auf dieser Seite.

#### Festplattendienstprogramm

Das Festplattendienstprogramm befindet sich im Ordner Dienstprogramme (Befehl + Shift + U). Mit diesem Programm lassen sich die Partitionen der SD-Karte löschen, neu anlegen und formatieren, damit die Karte wieder für den normalen Gebrauch (z.B. als Speicherkarte in einer Digitalkamera) eingesetzt werden kann.

#### Terminal

Das Terminal befindet sich ebenfalls im Ordner Dienstprogramme. Über das Terminal kann eine Secure Shell (SSH) Verbindung zum Raspberry Pi aufgebaut werden (vgl. Abbildung 15).

```

MacKlaus — pi@raspberrypi: ~ — ssh — 80x14
Last login: Sun May 12 16:40:19 on console
Klauss-Mac-mini:~ MacKlaus$ ssh pi@25.4.69.200
pi@25.4.69.200's password:
Linux raspberrypi 3.6.11+ #371 PREEMPT Thu Feb 7 16:31:35 GMT 2013 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun May 12 15:15:14 2013 from klauss-mac-mini.local
pi@raspberrypi ~ $

```

Abbildung 15: Mac OS Terminal

## XQuartz [19]

XQuartz ist ein Open Source Projekt, um die X11 Oberfläche für Mac OS X zu entwickeln. Um den Bildschirminhalt des Raspberry Pi auf einem Rechner unter Mac OS X zu importieren sind folgende Schritte notwendig.

Nach dem Herunterladen und Installieren von XQuartz müssen in den Einstellungen der Vollbildmodus und der Zugriff auf die Menüleiste im Vollbildmodus aktiviert werden. Ersteres ermöglicht den Zugriff auf Objekte, die am Schreibtisch des Raspberry Pi liegen, zweiteres blendet die XQuartz Menüleiste ein, wenn der Benutzer mit dem Mauspeil den oberen Bildschirmrand berührt. Mit der Tastenkombination Befehl + Wahl + A kann der Vollbildmodus ein- oder ausgeschaltet werden (vgl. Abbildung 16).

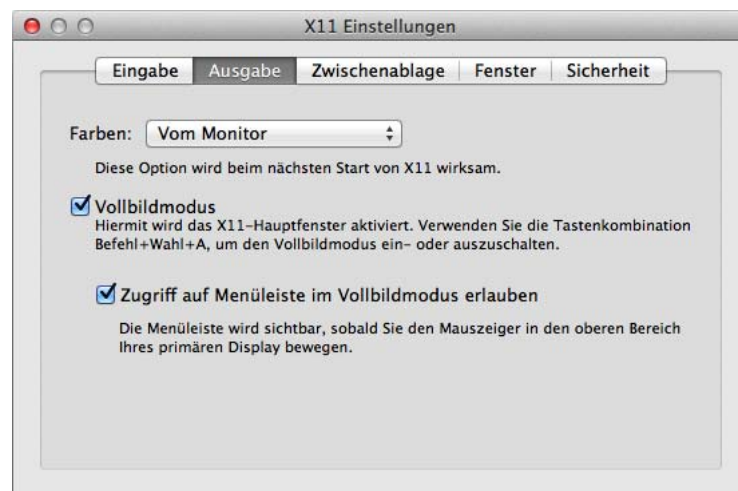


Abbildung 16: X11 Einstellungen

Zum Starten einer Session muss zunächst in einem Terminalfenster eine `ssh -X` Verbindung mit dem Raspberry Pi aufgebaut werden (z.B. `ssh -X pi@25.4.69.200`) und sich der Benutzer mit seinem Kennwort authentifizieren. Anschließend wird mit dem Befehl

```
lxsession&
```

der Bildschirminhalt des Raspberry Pi importiert.

## Netatalk

Um Dateien zwischen einem Rechner unter Mac OS X und dem Raspberry Pi auszutauschen, muss auf dem Raspberry Pi Netatalk installiert werden. Dazu sind die folgenden Konsolenbefehle abzusetzen.

Installieren von Netatalk:

```
pi@raspberrypi ~ $ sudo apt-get update
pi@raspberrypi ~ $ sudo apt-get install netatalk
```

Stoppen und Konfigurieren von Netatalk:

```
pi@raspberrypi ~ $ sudo /etc/init.d/netatalk stop
pi@raspberrypi ~ $ sudo nano /etc/netatalk/AppleVolumes.default
```

Am Ende der Textdatei *AppleVolumes.default* steht

```
~/ "Home Directory"
```

„~/“ ist das Home-Verzeichnis des angemeldeten Benutzers. Es können hier noch weitere Verzeichnisse hinzugefügt werden. Ein Eintrag um die Home-Verzeichnisse aller Schüler zu sehen wäre zum Beispiel:

```
/home „Rpi home“
```

Starten von Netatalk:

```
pi@raspberrypi ~ $ sudo /etc/init.d/netatalk start
```

Nach kurzer Zeit erscheint der Raspberry Pi unter den Freigaben in einem Mac OS X Finderfenster. Durch Klicken von „Verbinden als...“ und Eingabe von Benutzername und Kennwort werden die am Raspberry Pi freigegebenen Verzeichnisse sichtbar und können gelesen und beschrieben werden (vgl. Abbildung 17).

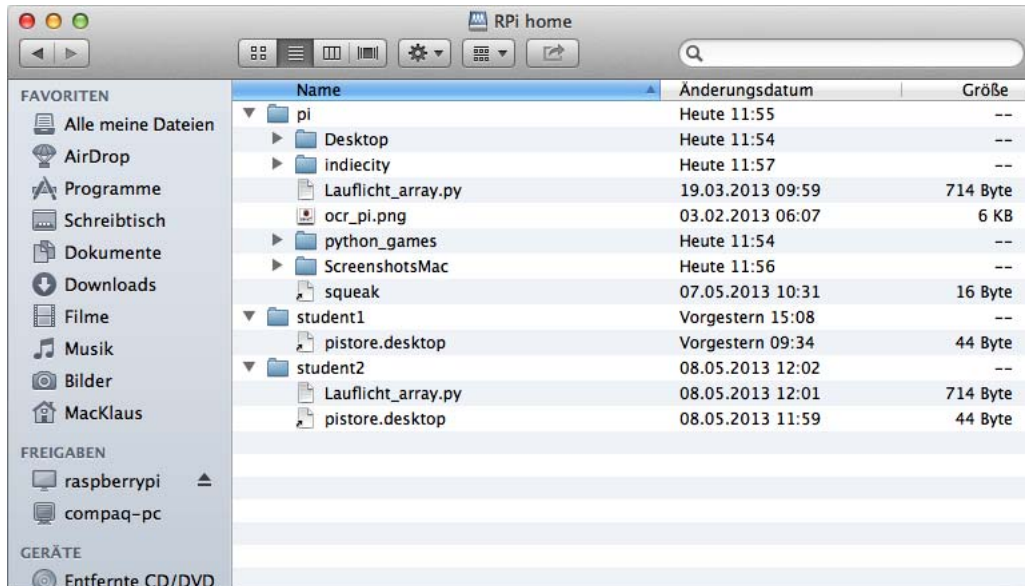


Abbildung 17: Freigabe unter Mac OS

## 4 Pidget Klassifikation

Wie in Kapitel 1.1 erwähnt, sind Pidgets elektronische Schaltungen (Widgets) für den Raspberry Pi (Pi + Widget = Pidget). Diese Pidgets werden an den General Purpose Input/Output (GPIO) Port angeschlossen und kommunizieren über diesen mit dem Raspberry Pi. In diesem Kapitel wird eine Möglichkeit zur Klassifizierung der Pidgets gezeigt. Dazu werden die Merkmale eines Pidgets und deren mögliche Ausprägungen angeführt und mit Beispielen veranschaulicht.

### 4.1 Interaktion mit der Umwelt

Dieses Merkmal unterteilt die Menge der Pidgets nach der Art wie sie mit dem Benutzer oder anderen Pidgets interagieren. Die Kommunikation mit dem Raspberry Pi ist immer elektronisch und wird hier nicht betrachtet (vgl. Tabelle 8).

Ausprägung	Beschreibung	Beispiel
Elektrooptisch	Sendet/Empfängt optische Signale	Angesteuerte LED, Signal eines Fototransistors
Elektromechanisch	Bewegt mechanische Teile	Motor, Servo
Elektroakustisch	Sendet/Empfängt akustische Signale	Piezosummer, Mikrofon
Elektronisch	Ausschließlich über elektrische Signale	A/D Wandler, Taster

Tabelle 8: Interaktionsausprägungen

### 4.2 Charakter

Damit ist die Art der Aufgaben gemeint, die für ein Pidget gestellt werden können (vgl. Tabelle 9). Ein Pidget kann mehrere der angeführten Ausprägungen des Merkmals besitzen. Die angeführten Beispiele finden sich im Kapitel 7.

Ausprägung	Beschreibung	Beispiel
Spielerisch	Spiel und Spaß stehen im Vordergrund.	Pi-Car: Fahrtrichtung umkehren
Technisch-praktisch	Die Aufgabe hat Technikbezug und es gibt praktische Anwendungen dazu	7-Segment Display: Stoppuhr Pi-Car: Geschwindigkeitsregelung
Innerinformatisch	Die Aufgabe vermittelt informatisches Grundlagenwissen	LED Strip: Darstellung von Binärzahlen

Tabelle 9: Charakterausprägungen

### 4.3 Fachzugehörigkeit

Dieses Merkmal beschreibt ob es für ein Pidget fächerübergreifende (FÜ) Aufgabenstellungen gibt. Die Liste der Ausprägungen kann bei Bedarf erweitert werden (vgl. Tabelle 10).

Ausprägung	Beschreibung	Beispiel
Informatik	Nur für das Fach Informatik	Optical Transmitter: Morsecodierung
FÜ Physik	Informatik + Physik	Analogwerterfassung mit A/D Wandler
FÜ Bildnerische E.	Informatik + BE	Magic Wand (Zauberstab) [20]
FÜ Sport	Informatik + Sport	7-Segment Display: Stoppuhr

Tabelle 10: Fachzugehörigkeitsausprägungen

#### 4.4 Kommunikation

Dieses Merkmal beschreibt die Kommunikation zwischen Raspberry Pi und dem Pidget. Diese kann uni- oder bidirektional (UD/BD) sein. Außerdem kann noch unterschieden werden, ob am GPIO einzelne Pins geschaltet werden bzw. deren Zustand gelesen wird (Kommunikation über einzelne Bits (EB)) oder ob die Kommunikation über einen Bus (beispielsweise dem Serial Peripheral Interface Bus, der auch am GPIO anliegt) erfolgt. Damit ergeben sich die folgenden vier Ausprägungen (vgl. Tabelle 11):

Ausprägung	Beschreibung	Beispiel
UD, EB	Kommunikation nur in eine Richtung und über einzelne Pins des GPIO	LED Strip
UD, Bus	Kommunikation nur in eine Richtung und über Bus	Analogwerterfassung
BD, EB	Kommunikation in beide Richtungen und über einzelne Pins des GPIO	7-Segment Display + Taster
BD, Bus	Kommunikation in beide Richtungen und über Bus	Zweizeiliges LC-Display mit Tasten[21]

Tabelle 11: Kommunikationsausprägungen

#### 4.5 Technische Realisierbarkeit

Die technische Realisierbarkeit klassifiziert ein Pidget nach Schwierigkeitsgrad und zeitlichem Aufwand um es aufzubauen.

Ausprägung	Beschreibung	Beispiel
fertig	Das Pidget gibt es fertig aufgebaut zu kaufen. Ein Nachbau ist nicht möglich oder lohnt sich nicht.	Zweizeiliges LC-Display mit Tasten
einfach, unaufwändig	Das Pidget lässt sich einfach und schnell aufbauen (weniger als eine Stunde Zeitaufwand).	LED Strip, D/A Wandler diskret aufgebaut
mittel	Der Schwierigkeitsgrad ist mittel und Zeitaufwand ist zwischen ein und zwei Stunden.	Optical Transmitter, Optical Receiver
schwierig, aufwändig	Die Schaltung ist komplex und/oder der Aufbau ist zeitaufwändig (mehr als zwei Stunden)	7-Segment Display
funktioniert nicht	Der Schwierigkeitsgrad ist nicht relevant, der Aufbau und/oder die Ansteuerung hat nicht funktioniert	Ansteuerung eines Digitalservos über Pulsweitenmodulation (PWM)

Tabelle 12: Realisierbarkeitsausprägungen

Die angegebenen Zeiten beziehen sich, so wie bei den in Kapitel 5 gezeigten Prototypen, auf einen Aufbau auf Lochrasterplatine mit Kupferlackdraht. Fertig geätzte Platinen sind natürlich in wesentlich kürzerer Zeit bestückt und gelötet.

#### 4.6 Verständlichkeit der Funktionsweise

Im Unterschied zur technischen Realisierbarkeit beschreibt dieses Merkmal wie einfach oder schwierig es für die Schüler ist, die elektronische Schaltung und ihre Funktion zu verstehen. Dieses Merkmal korreliert nicht immer mit der technischen Realisierbarkeit wie Tabelle 13 im Vergleich zu Tabelle 12 entnommen werden kann.



Ausprägung	Beschreibung	Beispiel
einfach	Simple Schaltung, Funktionsweise offensichtlich	LED Strip
mittel	Die Schaltung bedarf einer kurzen Erklärung benötigt aber keine tieferen Elektronikkenntnisse.	7-Segment Display
schwierig	Um die Funktionsweise zu verstehen, werden fundierte Elektronikkenntnisse benötigt	D/A Wandler diskret aufgebaut mit Operationsverstärker

Tabelle 13: Verständlichkeitsausprägungen

## 4.7 Stromverbrauch

Dieses Merkmal gibt den maximalen Stromverbrauch des Pidgets an (vgl. Kapitel 5). Der Wert ist für die richtige Wahl des Netzteiles von Bedeutung. Wie schon erwähnt, führen zu schwache Netzteile zu Schwankungen der Versorgungsspannung und in der Folge zu Instabilitäten des Raspberry Pi.

## 4.8 Preis

Mit diesem Merkmal werden die Kosten der für das Pidget verwendeten Bauteile beschrieben. Grundlage sind die Preise von RS-Components [7] vom Mai 2013. Die Preise der Pidgets werden in Kapitel 5 angegeben.

## 5 Realisierte Pidgets

Zu Beginn dieses Kapitels werden Überlegungen angeführt, die für die Auswahl der realisierten Pidgets ausschlaggebend waren. Auf Grundlage der Klassifizierungskriterien des letzten Kapitels wird diskutiert, welche Ausprägungen der Merkmale eines Pidgets für die Verwendung im Unterricht gut oder weniger gut geeignet sind.

Danach werden wichtige Informationen zum GPIO Port beschrieben. Diese beinhalten die Belegung der Pins, elektrische Limits des Ports sowie Erklärungen zum Initialzustand der Pins. Auch mögliche Fehlerquellen bei der Gestaltung von Pidgets, die zum Zerstören des Raspberry Pi und/oder des Pidgets führen können, werden angeführt.

Der Hauptteil des Kapitels widmet sich den für diese Arbeit aufgebauten Pidgets. Jedes der Pidgets wird nach der in Kapitel 4 gezeigten Einteilung klassifiziert. Dies dient dazu, um dem Leser einen raschen Überblick darüber zu geben, welche Merkmale das Pidget hat und wie es im Unterricht eingesetzt werden kann. Danach erfolgt die technische Dokumentation des Pidgets, d.h. Schaltplan, Bauteilliste und ggf. Hinweise zu Schaltungsvarianten und zur Funktionsweise.

Zum Abschluss des Kapitels folgt dann exemplarisch eine Schritt-für-Schritt Anleitung für den Aufbau eines Pidgets und für die Herstellung des zwischen Raspberry Pi und Pidget benötigten Verbindungskabels.

### 5.1 Kriterien

Auf Basis der in Kapitel 4 vorgestellten Merkmale eines Pidgets und deren mögliche Ausprägungen folgen in diesem Abschnitt Überlegungen, welche davon für den Unterrichtsgebrauch als günstig eingeschätzt werden.

#### 5.1.1 Interaktion mit der Umwelt

Für den Unterricht sind rein elektronische oder elektrooptische Pidgets besser geeignet als elektroakustische oder elektromechanische. Elektroakustische Pidgets erzeugen Geräusche, welche den Lärmpegel in der Klasse heben und die Schüler von der Arbeit ablenken. Pidgets mit mechanischen Komponenten sind für Defekte anfälliger. Sie sind auch von den Abmessungen größer und können (so mehrfach vorhanden) dadurch Probleme bei der Lagerung verursachen. Bei den realisierten Pidgets wurde darauf geachtet, dass die Abmessungen klein sind und dass auf vorstehende Teile möglichst verzichtet wird, sodass die Pidgets gestapelt und damit einfach gelagert werden können.

#### 5.1.2 Charakter

Hier sind bei den realisierten Pidgets alle Ausprägungen (spielerisch, technisch-praktisch, innerinformatisch) vorhanden – zum Teil in einem einzigen Pidget vereint. Für solche Pidgets sind daher viele verschiedene Aufgabenstellungen möglich und sie sind im Unterricht besser einsetzbar als Pidgets, die beispielsweise ausschließlich spielerischen Charakter haben.

#### 5.1.3 Fachzugehörigkeit

Die realisierten Pidgets sind hauptsächlich dem Informatikunterricht zugehörig. Der Grund dafür ist, dass sowohl der Aufbau technisch anspruchsvollerer Pidgets als auch die Planung von fächerübergreifendem Unterricht den Rahmen dieser Arbeit sprengen würde.

### 5.1.4 Kommunikation

Die Kommunikation zwischen Raspberry Pi und den Pidgets erfolgt ausschließlich bitweise und je nach Pidget uni- oder bidirektional. Bitweise Kommunikation ist für die Schüler einfacher zu verstehen und auch einfacher zu programmieren als die Kommunikation über einen Bus.

### 5.1.5 Technische Realisierbarkeit

Der Schwierigkeitsgrad und zeitliche Aufwand, um die Pidgets aufzubauen, ist mit einer Ausnahme (7-Segment Display) gering. Beim erwähnten Display war neben den verschiedenen Einsatzmöglichkeiten zugegebener Maßen auch eine Portion „Spaß an der Sache“ die Motivation dieses Pidget aufzubauen.

### 5.1.6 Verständlichkeit der Funktionsweise

Diese hat Vorrang gegenüber ausgefallenen Funktionen. Wie schon in Kapitel 0 erwähnt, sollen die Schüler die Funktion des Pidgets verstehen, damit über die Pidgets eine Verbindung zwischen Programm und realer Welt entstehen kann. Deshalb sind die Schaltungen so gewählt, dass ihre Funktion entweder offensichtlich ist, oder mit wenigen Worten erklärt werden kann. Die einzige Ausnahme stellt der Optical Receiver dar, bei dem ein Operationsverstärker zum Einsatz kommt. Ideal ist, wenn die Schüler ihre Pidgets im Zuge des Physik- oder des Werkunterrichts selbst herstellen können.

## 5.2 Technische Grundlagen zum GPIO Port

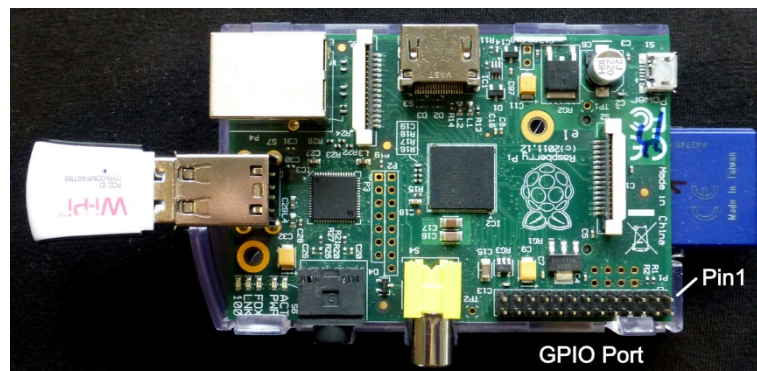


Abbildung 18: GPIO Port des Raspberry Pi

Der GPIO Port ist als 26-polige Stiftleiste (2-reihig, Stiftabstand 2,54mm) ausgeführt. 17 der 26 Pins können entweder als Ein- oder als Ausgang verwendet werden. Welche Funktion ein Pin hat, wird im Programm festgelegt, das den GPIO Port anspricht.

In Abbildung 18 ist der Raspberry Pi so angeordnet, dass sich Pin 1 der Stiftleiste rechts oben befindet. Die Pins sind so nummeriert, dass die Nummern von rechts nach links ansteigen. Pins mit ungerader Nummer befinden sich in der oberen Reihe der Stiftleiste, Pins mit gerader Nummer in der unteren. Abbildung 19 zeigt die Belegung der Pins des GPIO Ports:

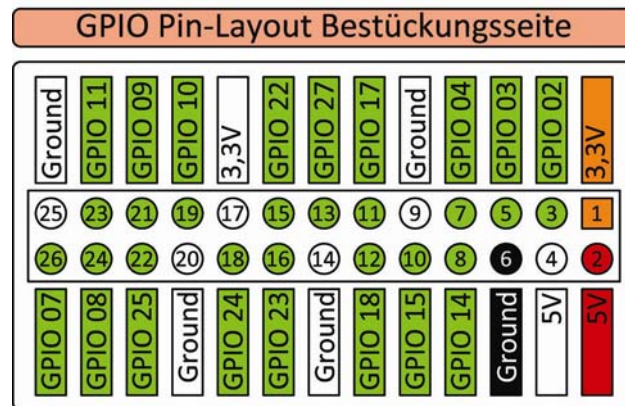


Abbildung 19: Pin-Belegung GPIO, Bestückungsseite

**Hinweise:**

- Um einen Pin des GPIO Ports anzusteuern gibt es in Python zwei Möglichkeiten, nämlich ihn entweder über die GPIO-Nummer (z.B. „GPIO 02“) oder über die Pin-Nummer (z.B. „3“) anzusprechen.  
Zwischen Revision 1 und Revision 2 der Raspberry Pi Boards wurde die Belegung der GPIO-Nummern geändert, die der Pin-Nummer ist jedoch gleich geblieben – so liegt bei Boards der Revision 1 beispielsweise an Pin 3 der „GPIO 00“ an, bei Boards der Revision 2 am selben Pin aber „GPIO 02“. Es wird daher aus Gründen der Kompatibilität zu den verschiedenen Modellen des Raspberry Pi in dieser Arbeit ausschließlich die Ansteuerung via Pin-Nummern verwendet.[22]
- Pins mit weißem Hintergrund (3,3V, 5V, Ground) sind zwar so belegt, wie sie beschriftet sind, könnten in späteren Versionen des Raspberry Pi aber noch anders belegt werden. Es wird daher empfohlen die Pins 1, 2 und 6 für Spannungsversorgungen mit 3,3 oder 5V bzw. für die Masse zu verwenden.
- Abbildung 19 zeigt die Pin-Belegung des GPIO Ports von der Oberseite des Raspberry Pi betrachtet (Bestückungsseite). Um beim Aufbau eines Pidgets Verdrahtungsfehler zu vermeiden, ist es meist zweckmäßiger, mit der Belegung des GPIO Ports an der Unterseite des Raspberry Pi zu arbeiten (Lötseite). Deshalb wird auch diese in Abbildung 20 angegeben:

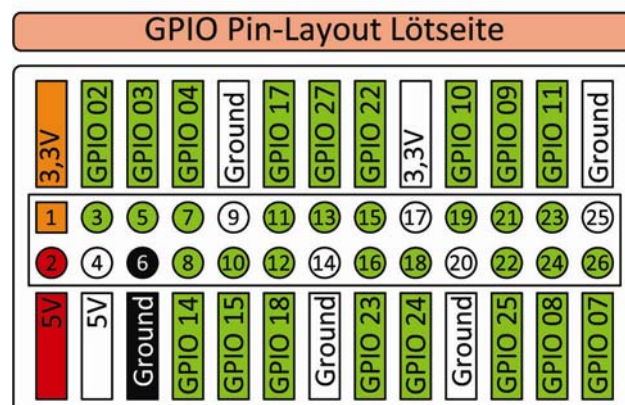


Abbildung 20: Pin-Belegung GPIO, Lötseite

### 5.2.1 Elektrische Grenzwerte

#### Warnung:

- Die Spannungen und Ströme an den Pins des GPIO Ports müssen innerhalb zulässiger Grenzen bleiben. Ein Überschreiten führt zu dauerhaften Schäden am Raspberry Pi!

Wie erwähnt kann jeder Pin des GPIO Ports entweder als Ein- oder als Ausgang konfiguriert werden. Ein Ausgangspin liefert eine Spannung von maximal 3,3V, die Spannung an einem Eingangspin darf diesen Wert ebenfalls nicht übersteigen. Der Strom über einen Pin (egal welche Richtung) darf nicht größer als 16mA sein. Für die Schwellspannungen zwischen den Zuständen „high“ und „low“ gelten folgende Werte:[23]

- Spannungen bis zu 0,8V werden von einem als Eingang konfigurierten GPIO Pin als Zustand „low“ interpretiert. Analog dazu ist die Spannung an einem ausgeschalteten Ausgangspin höchstens 0,8V.
- Spannungen von mindestens 1,3V werden von einem als Eingang konfigurierten GPIO Pin als Zustand „high“ interpretiert. Analog dazu ist die Spannung an einem eingeschalteten Ausgangspin mindestens 1,3V.
- Spannungen im Bereich größer 0,8V und kleiner 1,3V führen zu keinem eindeutigen Zustand.

Für die **Power-Pins** gelten folgende Grenzwerte: Pin 1 (3,3V Spannungsversorgung) darf mit maximal 50mA belastet werden. Pin 2 (5V Spannungsversorgung) ist mit dem Netzteil verbunden. Dieser Pin kann somit die Differenz zwischen dem maximalen Strom aus dem Netzteil und dem Stromverbrauch des Raspberry Pi liefern. Für ein 5V/1,5A Netzteil und einen Raspberry Pi Modell B2 wären das zum Beispiel  $1500 - 700 = 800\text{mA}$ .

### 5.2.2 Initialzustand der GPIO Pins

Wenn der Raspberry Pi gestartet wird, werden nur acht GPIO Pins als normale Ein- oder Ausgänge initialisiert. Die restlichen neun Pins werden so initialisiert, dass der Raspberry Pi über diese Pins mit drei verschiedenen Datenbussystemen kommunizieren kann. Tabelle 14 zeigt die Belegung dieser Pins:

I <sup>2</sup> C Bus		
Der Inter-Integrated Circuit Bus (I <sup>2</sup> C) dient hauptsächlich zur geräteinternen Kommunikation zwischen verschiedenen ICs über kurze Distanzen. Im Bus gibt es mindestens einen Master (Raspberry Pi) und maximal 127 Slaves.		
GPIO-Pin	Belegung	Beschreibung
3	SDA	Datenleitung des Busses
5	SCL	Taktleitung
UART		
Die Universal Asynchronous Receiver/Transmitter (UART) Schnittstelle dient zum Senden und Empfangen von Daten zwischen PCs und/oder Mikrocontrollern. Die Datenübertragung ist asynchron, daher ist kein Takt notwendig.		
GPIO-Pin	Belegung	Beschreibung
8	Receive	Empfangsleitung des Raspberry Pi
10	Transmit	Sendeleitung des Raspberry Pi
SPI		
Der Serial Peripheral Interface (SPI) Bus dient zur synchronen seriellen Datenübertragung nach dem Master-Slave-Prinzip mit einem Master und mehreren Slaves. Der Raspberry Pi unterstützt den Datenaustausch mit bis zu zwei Slaves. Ein Slave kann zum Beispiel ein Analog-Digital-Wandler sein.		
GPIO-Pin	Belegung	Beschreibung
19	MOSI	Master Out Slave In: Sendeleitung des Raspberry Pi

21	MISO	Master In Slave Out: Empfangsleitung des Raspberry Pi
23	SCLK	Taktleitung
24	CE0	Chip Enable erster Slave
26	CE1	Chip Enable zweiter Slave

Tabelle 14: Initialzustände der GPIO Pins

Für alle 17 GPIO Pins gilt, dass ihre Zustände in einem Python Programm neu gesetzt werden können. Die Initialisierung wirkt sich daher bei der Ausführung des Programms nicht mehr aus. Wichtig ist allerdings, dass ein angeschlossenes Pidget in der Startphase des Raspberry Pi keine Überlastungen einzelner Pins verursacht, was zu Schäden am Raspberry Pi führen würde. Im Folgenden werden Möglichkeiten gezeigt um solche Überlastungen zu vermeiden.

### 5.2.3 Designhinweise

Eine Schwierigkeit beim Aufbau von Pidget-Schaltungen ist die, dass GPIO Pins in der Software als Ein- und auch als Ausgang konfiguriert werden können und Ausgangspins die Zustände „high“ und „low“ einnehmen. Im Unterricht wird es Schülern passieren, dass sie Pins falsch konfigurieren, was aber weder Schäden am Raspberry Pi noch am Pidget nach sich ziehen darf. Aus diesem Grund wird hier auf zwei mögliche Gestaltungsfehler hingewiesen.

Schaltung	Problembeschreibung	Verbesserte Schaltung
	<p>Die links abgebildete Schaltung funktioniert, solange der Pin als Eingangspin konfiguriert ist. Am Eingang des GPIO Pins liegt das invertierte Signal des Schalters an.</p> <p>Wird der Pin allerdings irrtümlich als Ausgang definiert und auf „high“ gesetzt, dann führt das Schließen des Schalters zu einem Kurzschluss und damit zu einer Beschädigung des Raspberry Pi.</p> <p>Abhilfe schafft ein zusätzlicher Serienwiderstand <math>R_S</math> (z.B.: <math>1000\Omega</math>), der bei Falschkonfiguration den Strom auf ein zulässiges Maß reduziert.</p>	
	<p>Die links abgebildete Schaltung funktioniert ebenfalls solange der Pin als Eingangspin konfiguriert ist. Hier liegt am Eingang des GPIO Pins das Schaltersignal an.</p> <p>Wird allerdings der Pin irrtümlich als Ausgang definiert und auf „low“ gesetzt, dann führt auch hier das Schließen des Schalters zu einem Kurzschluss und einer Beschädigung des Raspberry Pi. Der in der rechten Schaltung eingefügte Serienwiderstand <math>R_S</math> (z.B. <math>1000\Omega</math>) verhindert das.</p>	

Tabelle 15: Mögliche Gestaltungsfehler

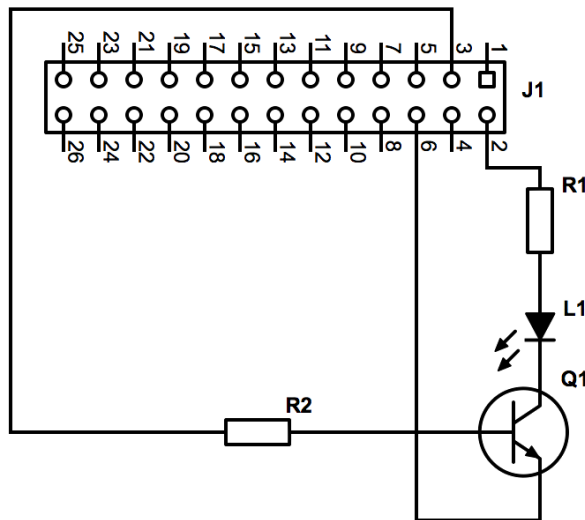


Abbildung 21: Schaltungsvariante

Für Verbraucher, die einen GPIO Pin mit mehr als den zulässigen 16mA belasten würden, kann die in Abbildung 21 abgebildete Schaltungsvariante verwendet werden. Hier wird der Verbraucher über Pin 2 mit 5V versorgt und über einen Transistor geschaltet. Der GPIO Pin wird nur mit dem Basisstrom des Transistors belastet.

Wenn  $R_2$  beispielsweise  $10000\Omega$  gewählt wird, so reduziert sich der Strom über Pin 3 auf:

$$I = \frac{U_{\text{Pin3}}}{R_2} = \frac{3,3}{10000} = 0,33\text{mA}$$

Ein weiteres Problem kann darin bestehen, dass eine Schaltung eine Spannung ausgibt, die höher als die erlaubten 3,3V ist. In diesem Fall kann die Ausgangsspannung über einen Spannungsteiler (vgl. Abbildung 22) auf den erlaubten Wert von 3,3V gebracht werden. Eine (optionale) 3,3V Zenerdiode in Sperrrichtung schützt den GPIO Pin zusätzlich vor Überspannungen.

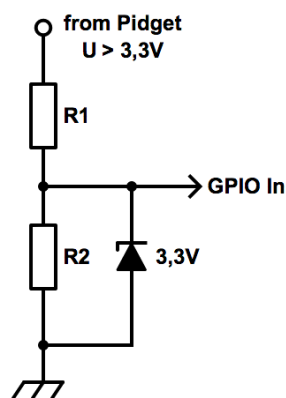


Abbildung 22: Spannungsteiler

Die Dimensionierung des Widerstandes  $R_2$  erfolgt nach folgender Formel (unbelasteter Spannungsteiler, Ausgangsspannung 3,3V):

$$R_2 = R_1 \frac{3,3}{U - 3,3}$$

Wobei  $U$  die Spannung aus der Schaltung ist ( $>3,3\text{V}$ ) und  $R_1$  angenommen werden muss.

**Beispiel:** die Spannung aus Pidget ist 5V,  $R_1=1000\Omega$

$$R_2 = R_1 \frac{3,3}{U - 3,3} = 1000 \frac{3,3}{5 - 3,3} = 1941\Omega$$

Ausgehend vom rechnerischen Wert muss nun der nächstkleinere Wert aus einer Widerstandsreihe gewählt werden. Für dieses Beispiel ist damit  $R_2=1800\Omega$ .

Als letzter Punkt soll noch die Kombination von GPIO (Logikpegel 3,3V) und TTL (Transistor-Transistor-Logik, Logikpegel 5V) betrachtet werden. Hier bietet die Firma Sparkfun [24] einen Logic Level Converter an (3,3V↔5V), der die Spannungspegel anpasst.

### 5.3 Dokumentation Pidgets

In diesem Abschnitt werden die realisierten Pidgets dokumentiert. Jedes Pidget wird verbal kurz beschrieben und klassifiziert. Neben dem Schaltplan wird auch eine Stückliste angegeben. Die angeführten Bestell-Nummern beziehen sich auf den Online Shop von RS-Components [7], ebenso der angegebene Gesamtpreis für das Pidget. Gehäuse und Frontplatten sind in die angegebenen Preise nicht mit eingerechnet.

Wo es erforderlich erscheint, werden noch zusätzliche Informationen über die Funktion oder Hinweise zum Aufbau des Pidgets angegeben.

### 5.3.1 Pi-Car

Ein Pi-Car ist ein funkferngesteuertes Modellauto dessen Fernsteuerung modifiziert und an den GPIO Port des Raspberry Pi angeschlossen wird. Ein Programm am Raspberry Pi steuert dann die Fernsteuerung, die ihrerseits das Modellauto steuert. Pi-Cars können als Tool Kit oder bereits fertig umgebaut bestellt werden. [25]

Das für diese Arbeit selbst zum Pi-Car umgebaute Modellauto war ein Vorversuch um die Funktionsweise und Programmierung des GPIO Ports kennen zu lernen und ist im eigentlichen Sinne kein Pidget.

Klassifikation Pi-Car	
Interaktion mit der Umwelt	Elektromechanisch
Charakter	Spielerisch Technisch-praktisch
Fachzugehörigkeit	Informatik FÜ technisches Werken FÜ Physik
Kommunikation	UD, EB
Technische Realisierbarkeit	einfach, unaufwändig
Verständlichkeit der Funktionsweise	einfach
Stromverbrauch	10 mA
Preis	2,06 € (zuzüglich Preis für Modellauto: 14,99€)

Tabelle 16: Klassifikation Pi-Car

Das Pi-Car hat hauptsächlich spielerischen Charakter. Nachdem es nicht nur über ein Python Programm angesprochen werden kann, sondern auch aus Scratch (vgl. [26]) ist es für Unterstufenschüler ein – im wahrsten Sinne des Wortes – geeignetes Vehikel um erste Erfahrungen in der Programmierung zu sammeln.

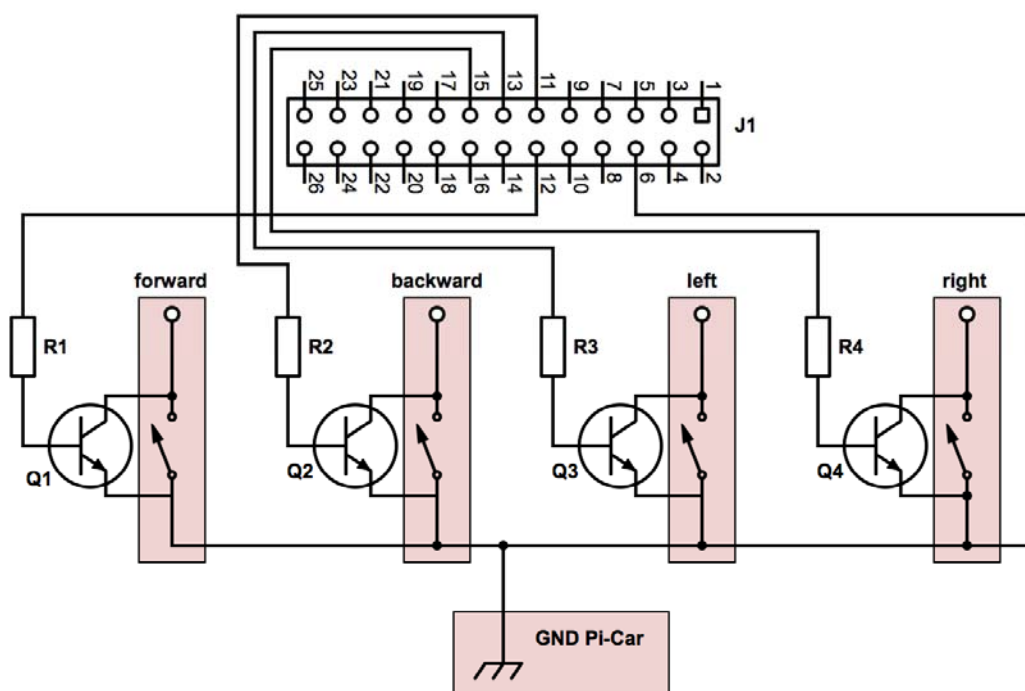


Abbildung 23: Schaltplan Pi-Car



Abbildung 23 zeigt den Schaltplan für den Umbau der Fernsteuerung. Die rosa hinterlegten Elemente sind die vier Tasten für die Bewegungen des Modellautos (vor, zurück, links, rechts) und die Masse der Fernsteuerung.

In Tabelle 17 sind die Bauteile für das Pi-Car aufgelistet:

Bauteil	Wert/Bezeichnung	BestellNr.
R <sub>1</sub> -R <sub>4</sub>	1000Ω, 0,25W	7077666
Q <sub>1</sub> -Q <sub>4</sub>	BC546	5449292A
J <sub>1</sub>	Steckverbinder	6741249

Tabelle 17: Stückliste Pi-Car

Die Transistoren Q1 bis Q4 werden vom GPIO Port angesteuert und erfüllen dieselbe Funktion wie ein Tastendruck auf der Fernsteuerung. Abbildung 24 zeigt die Fernsteuerung im geschlossenen und geöffneten Zustand.



Abbildung 24: Fernsteuerung geschlossen - geöffnet

Es sind die Taster für vor, zurück, links und rechts zu erkennen. Im rechten Teilbild wurden die Leitungen, die zu den Tastern führen, rot markiert und die Masseleitung blau. An diese Leitungen muss das Kabel der Elektronik des Pi-Cars angeschlossen werden. Abbildung 25 zeigt dies.

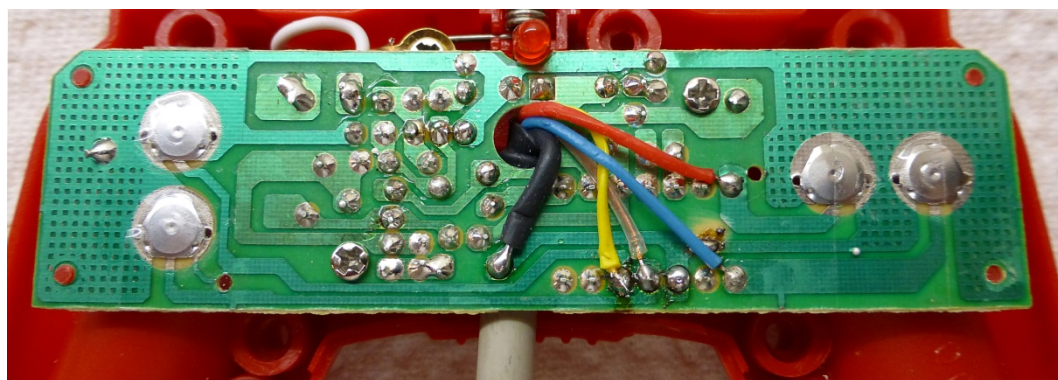


Abbildung 25: Kabel angeschlossen

Zusammgebaut sieht die Fernbedienung samt Pi-Car Elektronik und Verbindungskabel zum GPIO Port so wie in Abbildung 26 dargestellt aus.



Abbildung 26: Fertig umgebaute Fernbedienung

### 5.3.2 LED Strip

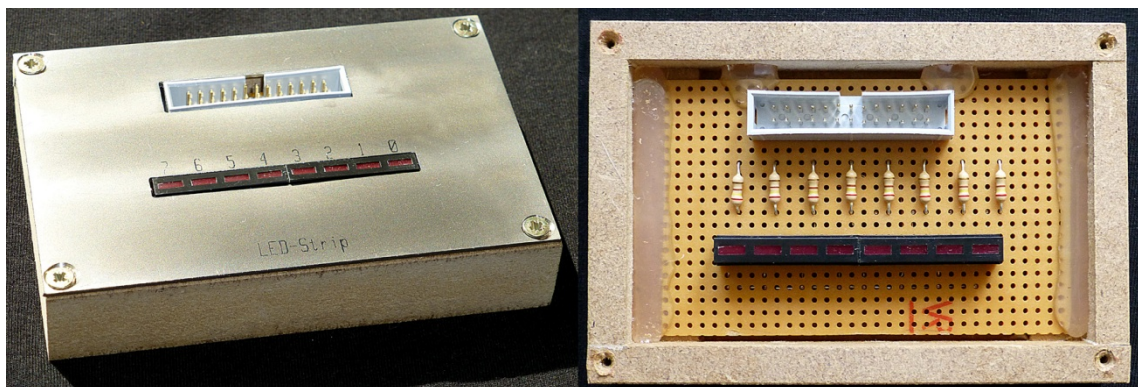


Abbildung 27: Außen- und Innenansicht LED Strip

Der LED Strip besteht aus einer Reihe von acht LEDs, die vom GPIO Port einzeln angesteuert werden können (vgl. Abbildung 27).

Klassifikation LED Strip	
Interaktion mit der Umwelt	Elektrooptisch
Charakter	Spielerisch Technisch-praktisch Innerinformatisch
Fachzugehörigkeit	Informatik
Kommunikation	UD, EB
Technische Realisierbarkeit	einfach, unaufwändig
Verständlichkeit der Funktionsweise	einfach
Stromverbrauch	48 mA
Preis	4,71 €

Tabelle 18: Klassifikation LED Strip

Der LED Strip ist ein Pidget mit vielen Einsatzmöglichkeiten, so können dafür spielerische Aufgaben gestellt werden (z.B. ein Lauflicht), Aufgaben mit praktischem Hintergrund (Hel-

lichtigkeitsregulierung über Pulsweitenmodulation) oder auch rein informatische Aufgaben (Darstellung von Binärzahlen). Abbildung 28 zeigt den Schaltplan des Pidgets.

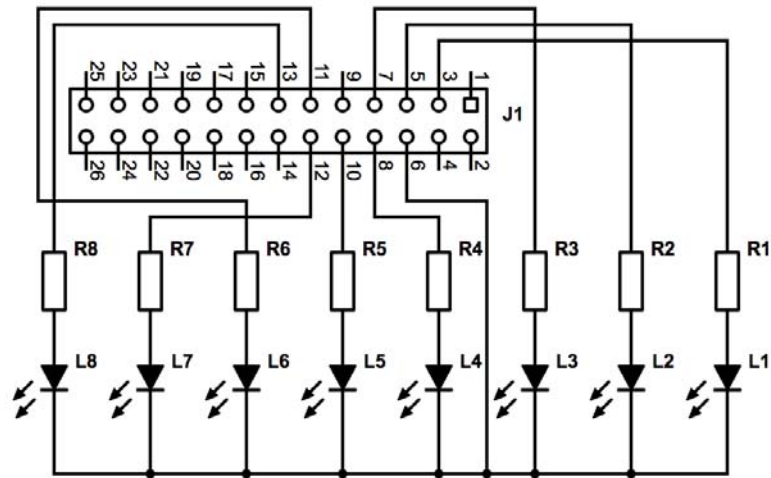


Abbildung 28: Schaltplan LED Strip

In Tabelle 19 sind die Bauteile für das LED Strip Pidget aufgelistet. Beim Prototyp bestehen die acht Leuchtdioden aus zwei LED-Segmenten zu je vier Leuchtdioden. Es können aber auch acht einzelne LEDs verwendet werden.

Bauteil	Wert/Bezeichnung	BestellNr.
R <sub>1</sub> -R <sub>8</sub>	240Ω, 0,25W	7077616
D <sub>1</sub> -D <sub>4</sub>	LED 4-Segment	2473422
D <sub>5</sub> -D <sub>8</sub>		
J <sub>1</sub>	Steckverbinder	6741249

Tabelle 19: Stückliste LED Strip

### 5.3.3 7-Segment Anzeige

Dieses Pidget besteht aus einer vierstelligen 7-Segment Anzeige und einem Taster. Die vier Stellen der Anzeige können unabhängig voneinander angesteuert werden. Dieses Pidget ist im Vergleich zu den anderen viel aufwändiger zu bauen, wie die Rückansicht der Lochrasterplatte in Abbildung 29 zeigt.

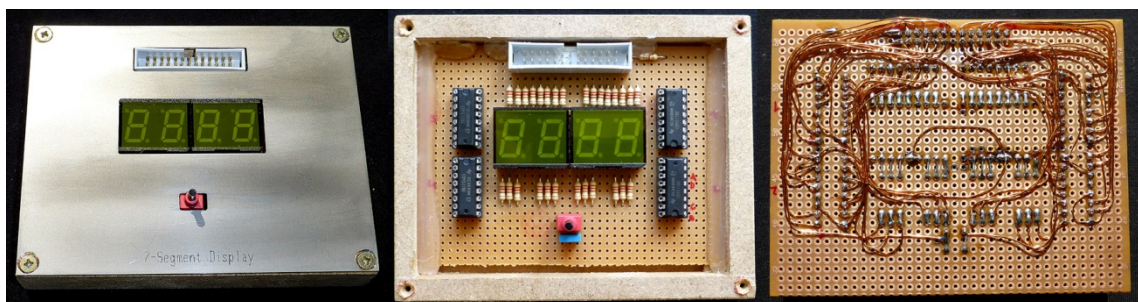


Abbildung 29: Außen-, Innen- und Rückansicht 7-Segment Display

Klassifikation 7-Segment Display	
Interaktion mit der Umwelt	Elektrooptisch
Charakter	Spielerisch Technisch-praktisch
Fachzugehörigkeit	Informatik
Kommunikation	BD, EB

Technische Realisierbarkeit	schwierig, aufwändig
Verständlichkeit der Funktionsweise	mittel
Stromverbrauch	280 mA
Preis	13,76 € (+0,38 € für Pull-Down Widerstände)

Tabelle 20: Klassifikation 7-Segment Display

Das Herzstück der Schaltung ist IC<sub>1</sub> (CD4511, BCD-to-7-Segment Decoder). Dieser Baustein hat vier Eingänge (A bis D), an denen die darzustellende Ziffer binär anliegt. Der IC schaltet die Ausgänge (a bis g) so, dass auf einer angeschlossenen 7-Segment Anzeige die Ziffer angezeigt wird. Wird an den Eingängen eine Zahl größer als neun (Dezimal) angelegt, so schaltet der IC die Anzeige ab. Abbildung 30 zeigt den Schaltplan für die erste Stelle des Pidgets. Die restlichen drei Stellen sind analog dazu aufzubauen. Die Zuordnung der GPIO Pins zu den Eingängen A-D der vier ICs wird in Tabelle 22 aufgelistet. Die Nummerierung der Stellen erfolgt von rechts nach links.

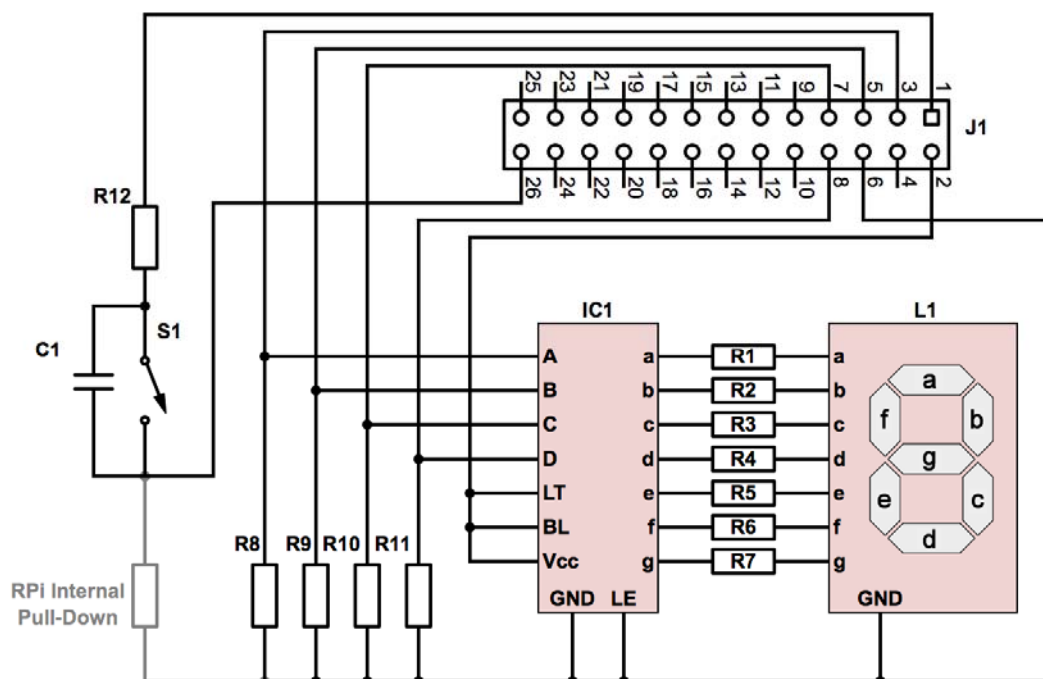


Abbildung 30: Schaltplan 7-Segment Display

In Tabelle 21 sind die Bauteile für eine Stelle der 7-Segment Anzeige aufgelistet. Mit Ausnahme von R<sub>12</sub>, S<sub>1</sub>, C<sub>1</sub> und J<sub>1</sub> werden für das gesamte Pidget die Bauteile in vierfacher Menge benötigt.

Bauteil	Wert/Bezeichnung	BestellNr.
R <sub>1</sub> -R <sub>7</sub>	220Ω, 0,25W	7077612
R <sub>8</sub> -R <sub>11</sub>	12kΩ, 0,25W	7077757
R <sub>12</sub>	1800Ω, 0,25W	7077688
IC <sub>1</sub>	CD4511	6629698
	IC-Sockel 16 Pin	6742485
L <sub>1</sub>	LED Display	7085225
S <sub>1</sub>	Switch	7023505
C <sub>1</sub>	22nF, 100V	3121447
J <sub>1</sub>	Steckverbinder	6741249

Tabelle 21: Stückliste 7-Segment Display

Für die Verwendung des Tasters ist es wichtig, dass im Python-Programm bei GPIO Pin 26 der interne Pull-Down Widerstand aktiviert wird (vgl. Kapitel 6.3). Wie später bei der Aufbauanleitung des Dice Pidgets noch ausführlicher erklärt wird, dient der Kondensator  $C_1$  zum Entprellen des Tasters. Die Pull-Down Widerstände  $R_8 - R_{11}$  können weggelassen werden. Sie bewirken nur, dass das 7-Segment Display beim Einschalten des Raspberry Pi einen definierten Zustand hat und „0000“ anzeigt. Sobald die GPIO Pins initialisiert sind, haben die Widerstände  $R_8 - R_{11}$  auf die Funktion des Pidgets keinen Einfluss mehr.

Stelle	1. Stelle				2. Stelle				3. Stelle				4. Stelle			
Bit	A1	B1	C1	D1	A2	B2	C2	D2	A3	B3	C3	D3	A4	B4	C4	D4
GPIO Pin	3	5	7	8	10	11	12	13	15	16	18	19	21	22	23	24

Tabelle 22: Zuordnung GPIO Pins zu Treiber IC

### 5.3.4 Optical Transmitter

Dieses Pidget steuert einen Laserpointer an und kann mit geeigneter Kodierung (zum Beispiel Morsecode) Daten auf optischem Weg übertragen (vgl. Abbildung 31). Auf der Gegenseite werden die Lichtimpulse von einem weiteren Pidget (Optical Receiver, vgl. Kapitel 5.3.5) empfangen und wieder in elektrische Signale umgewandelt.

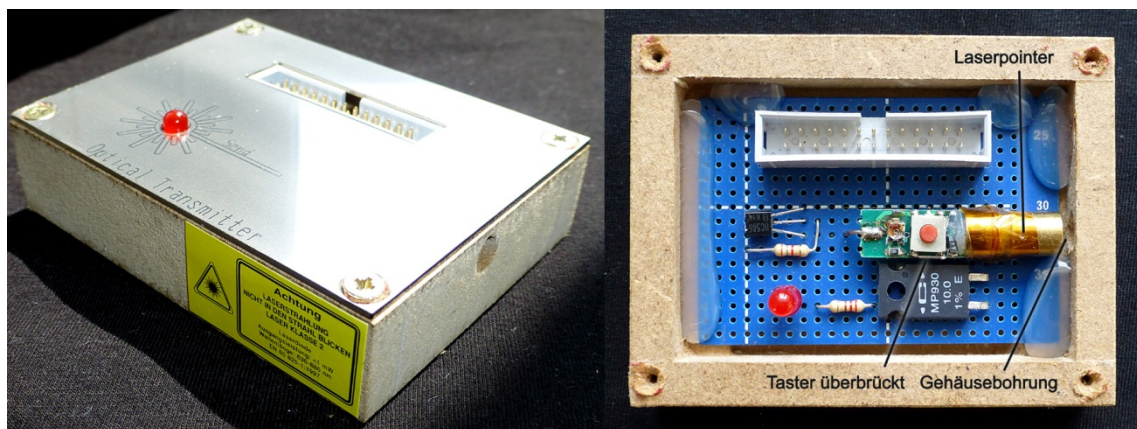


Abbildung 31: Außen- und Innenansicht Optical Transmitter

Klassifikation Optical Transmitter	
Interaktion mit der Umwelt	Elektrooptisch
Charakter	Technisch-praktisch
Fachzugehörigkeit	Informatik
Kommunikation	UD, EB
Technische Realisierbarkeit	mittel
Verständlichkeit der Funktionsweise	einfach
Stromverbrauch	10 mA
Preis	2,13 €, (+5,99 € für Laserpointer; z.B. Conrad Electronic [27])

Tabelle 23: Klassifikation Optical Transmitter

Bei diesem Pidget schaltet ein Transistor den Laserpointer und parallel dazu eine Leuchtdiode ein und aus. Der Laserpointer wurde für den Aufbau des Pidgets zersägt und nur das Innenleben (Laserdiode und Ansterelektronik) auf die Platine gesetzt. Der Taster am Laserpointer wurde überbrückt. Die Leuchtdiode hat den Zweck die Aktivität des Lasers beobachten zu können, ohne dabei die Augen zu gefährden.

**Warnung:**

- Laserstrahlung kann irreparable Schäden an Netzhaut, Hornhaut oder der Linse verursachen!
- Die Laserklassifizierung ist in der Lasersicherheitsnorm ÖNORM ÖVE EN 60825-1 festgelegt. Ungefährliche Laser werden der Klasse 1, die gefährlichsten Laser werden der Klasse 4 zugeordnet.

Für Lehrzwecke dürfen nur Laserquellen bis zur Klasse 2 (Wellenlängenbereich 400–700nm, Ausgangsleistung maximal 1mW) eingesetzt werden. Der Schutz des Auges wird bei diesen niedrigen Leistungen durch natürliche Abwendreaktionen (z.B. Lid-schlussreflex) gewährleistet. [28]

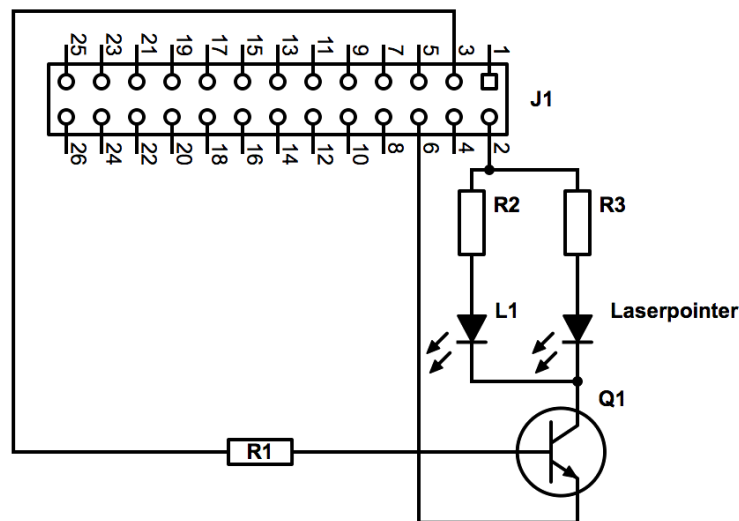


Abbildung 32: Schaltplan Optical Transmitter

In Tabelle 24 sind die Bauteile für den Optical Transmitter aufgelistet. Als Laserpointer fungierte ein Produkt der Firma Conrad Electronic. [27]

Bauteil	Wert/Bezeichnung	BestellNr.
R <sub>1</sub>	1800Ω, 0,25W	7077688
R <sub>2</sub>	220Ω, 0,25W	7077612
R <sub>3</sub>	10Ω, 0,25W	7077509
Q <sub>1</sub>	CD4511	6629698
L <sub>1</sub>	LED Display	7085225
Laserpointer		

Tabelle 24: Stückliste Optical Transmitter

Die Kombination Optical Transmitter und Optical Receiver wurde für das Verfassen dieser Arbeit gebaut und getestet. Es konnten dabei Impulsbreiten von mindestens 10ms aufgelöst werden. Die größte Distanz zwischen Transmitter und Receiver betrug im Test ca. 20m. Eine größere Distanz für die Datenübertragung ist vorstellbar, wurde jedoch nicht getestet. Jedes der Pidgets benötigt einen eigenen Raspberry Pi.

### 5.3.5 Optical Receiver

Der Optical Receiver (vgl. Abbildung 33) ist das Gegenstück zum Optical Transmitter. Dieses Pidget empfängt Lichtimpulse mit einem Fototransistor und setzt sie in elektrische Signale um.

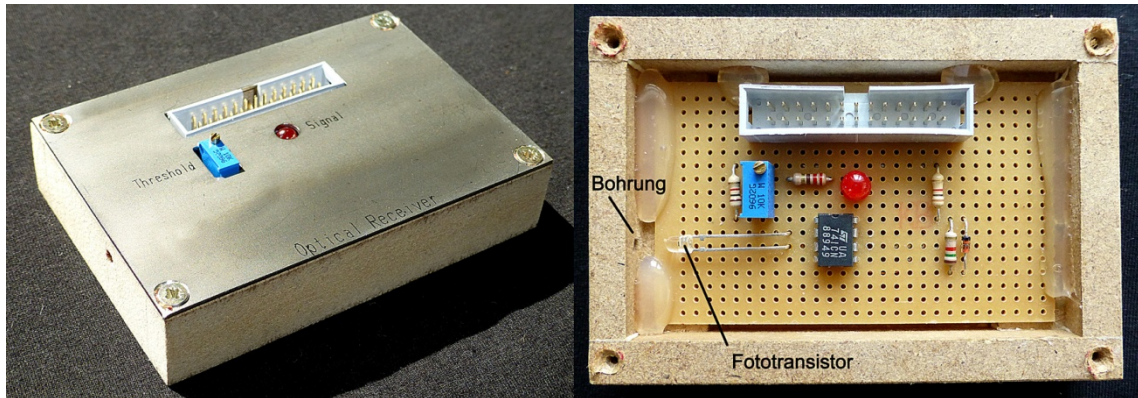


Abbildung 33: Außen- und Innenansicht Optical Receiver

Klassifikation Optical Receiver	
Interaktion mit der Umwelt	Elektrooptisch
Charakter	Technisch-praktisch
Fachzugehörigkeit	Informatik
Kommunikation	UD, EB
Technische Realisierbarkeit	mittel
Verständlichkeit der Funktionsweise	schwierig
Stromverbrauch	10 mA
Preis	4,45 €

Tabelle 25: Klassifikation Optical Receiver

Der Operationsverstärker IC<sub>1</sub> (vgl. Abbildung 34) wird als nicht invertierender Komparator betrieben. Das bedeutet, dass die Ausgangsspannung auf 5V steigt, sobald die Spannung am positiven Eingang höher ist, als jene am negativen und auf 0V im umgekehrten Fall.

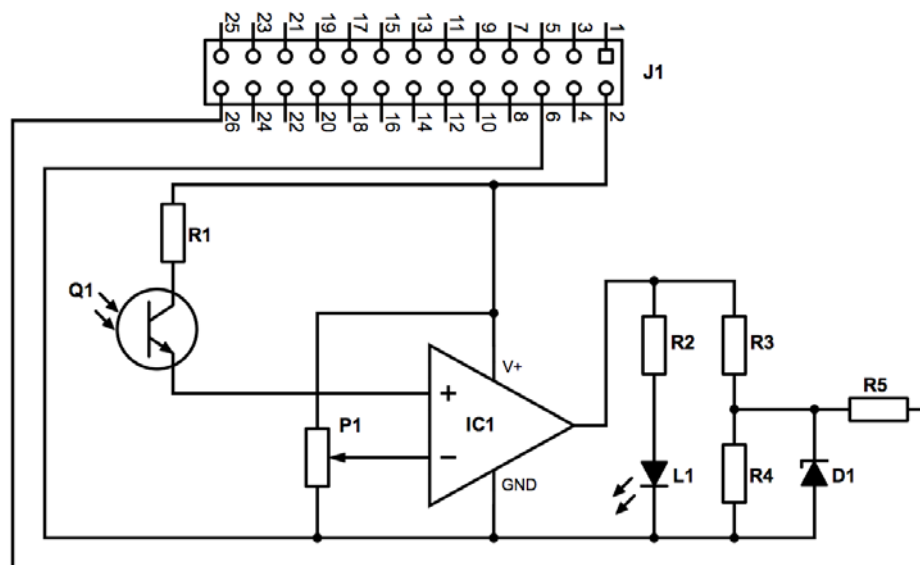


Abbildung 34: Schaltplan Optical Receiver

Die Spannung am positiven Eingang ändert sich mit der Helligkeit am Fototransistor. Die Spannung am negativen Eingang kann mit  $P_1$  zwischen 0-5V eingestellt werden und bestimmt die Schaltschwelle zwischen hell („ein“) und dunkel („aus“).

Die Leuchtdiode  $L_1$  dient nur zur Visualisierung des Zustands des Komparators und kann bei der Inbetriebnahme zum Abgleichen von  $P_1$  verwendet werden. Später dient diese LED den Schülern einerseits zur Kontrolle, ob ein optisches Signal empfangen wird und andererseits zum Überprüfen, ob das Programm die Signale am Pin 26 des GPIO Ports richtig auswertet.

#### Hinweise:

- Der in Abbildung 33 gezeigte Prototyp des Optical Receiver ist mit einem LM741 Operationsverstärker aufgebaut. Dieser hat den Nachteil, dass er nicht „Rail-to-Rail“ ist. Das bedeutet, dass die maximale Ausgangsspannung kleiner ist als die Versorgungsspannung und die minimale Ausgangsspannung höher als 0V. Deshalb musste beim Prototyp im Spannungsteiler der Widerstand  $R_4$  verkleinert werden und der Schutzwiderstand  $R_5$  entfallen. Da die minimale Ausgangsspannung höher als 0V ist, hat dies ein ständiges „Glimmen“ der LED zur Folge.
- In der in Tabelle 26 angegebenen Stückliste ist deshalb der Operationsverstärker durch einen LM358 ersetzt und  $R_4$  für einen Spannungsteiler für 5V dimensioniert worden.

Bauteil	Wert/Bezeichnung	BestellNr.
$R_1, R_3, R_5$	1000 $\Omega$ , 0,25W	7077666
$R_2$	220 $\Omega$ , 0,25W	7077612
$R_4$	1800 $\Omega$ , 0,25W	7077688
$IC_1$	LM358 Op.Amp. IC-Sockel 8 Pin	7613546 6742479
$D_1$	Zenerdiode	5443531
$L_1$	LED, rot, 5mm	2285988
$P_1$	10k $\Omega$ , Trimpot.	7692167
$Q_1$	Fototransistor	7082686
$J_1$	Steckverbinder	6741249

Tabelle 26: Stückliste Optical Receiver

### 5.3.6 Digitalservo

Dieser Abschnitt beschreibt – den leider nur teilweise erfolgreichen Versuch – ein Modellbau Digitalservo mit dem Raspberry Pi anzusteuern.

Abbildung 35 zeigt ein solches Servo. Die drei Anschlussleitungen sind braun (Masse), rot (+5V) und orange (Steuerimpulse). Das Impulsdiagramm für die Steuerimpulse ist in Abbildung 36 dargestellt.



Abbildung 35: Digitalservo

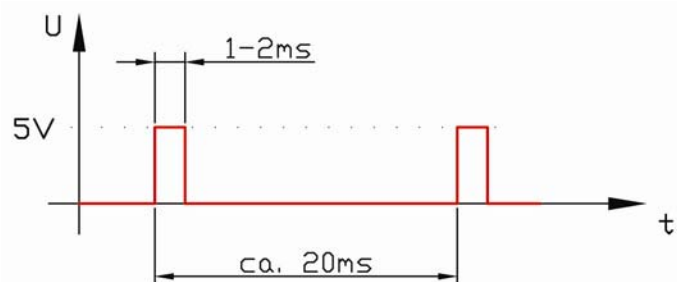


Abbildung 36: Impulsdiagramm

Die Information auf welche Position das Servo fahren soll, steckt in der Impulsbreite. Diese muss für den gesamten Verfahrbereich des Servos zwischen einer und zwei Millisekunden



sein. Zwischen zwei Impulsen ist eine Pause von ca. 20ms. Diese Dauer ist verglichen zur Impulsbreite unkritisch. Um ein solches Servo anzusteuern, wurde die in Abbildung 37 gezeigte Schaltung aufgebaut.

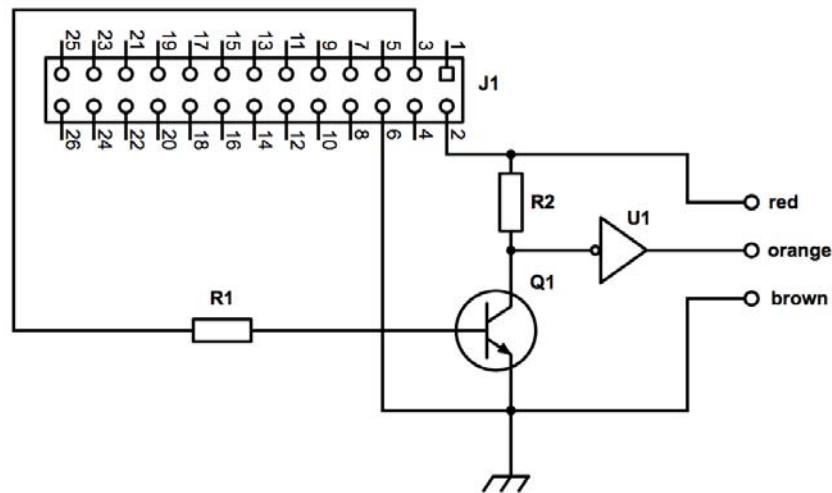


Abbildung 37: Ansteuerung Digitalservo

Der Transistor  $Q_1$  (BC546) und die Widerstände  $R_1$  und  $R_2$  (jeweils  $2700\Omega$ ) bilden einen Inverter, sodass am Eingang des Nicht-Gatters  $U_1$  (SN7404) das invertierte Signal von GPIO Pin 3 anliegt. Das Gatter invertiert das Signal noch einmal und hebt den Logikpegel auf 5V.

Das Python RPi.GPIO Paket bietet in der beim Erstellen dieser Arbeit verfügbaren Version 0.5.2a eine Software PWM (Pulsweitenmodulation), die für die Ansteuerung des Servos versuchsweise verwendet wurde. Diese PWM funktioniert grundsätzlich, allerdings ist das Timing zu instabil, sodass der Servo um die Sollposition „herumwackelt“.

In künftigen Versionen der RPi.GPIO Paketes soll Hardware PWM unterstützt werden und es sollte dann möglich sein, Digitalservos präzise mit dem GPIO Port anzusteuern. Damit würden sich dann die Möglichkeit eröffnen, den Raspberry Pi auf ein Fahrzeug oder einen Roboter zu setzen und dieses/diesen dann über WLAN fernzusteuern. [29]

## 5.4 Bauanleitung Dice Pidget und Verbindungskabel

Dieser Abschnitt zeigt anhand des Dice Pidgets, wie ein Pidget aufgebaut werden kann.

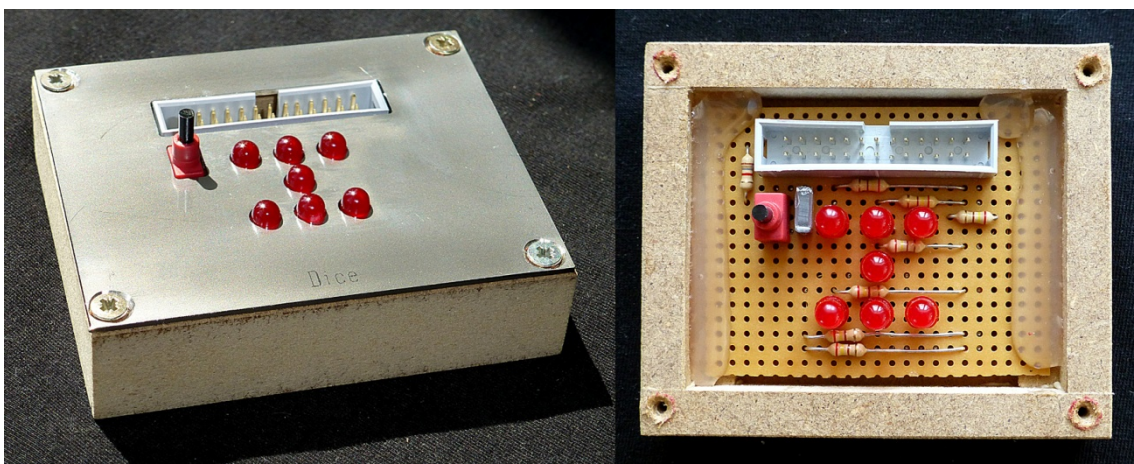


Abbildung 38: Außen- und Innenansicht Dice

Es wird dabei vorausgesetzt, dass der Umgang mit einem Lötkolben und Digitalmultimeter (zum Überprüfen der aufgebauten Schaltung) bekannt ist.

Das Dice Pidget besteht aus sieben Leuchtdioden (LEDs, vgl. Abbildung 38), die in der Form der Augen eines Spielwürfels angeordnet sind. Außerdem ist es noch mit einem Taster ausgestattet, mit dem zum Beispiel neu gewürfelt werden kann.

Klassifikation Dice	
Interaktion mit der Umwelt	Elektrooptisch
Charakter	Spielerisch
Fachzugehörigkeit	Informatik
Kommunikation	BD, EB
Technische Realisierbarkeit	einfach, unaufwändig
Verständlichkeit der Funktionsweise	einfach
Stromverbrauch	42 mA
Preis	5,11 €

Tabelle 27: Klassifikation Dice

### 5.4.1 Schaltplan und Funktion

Abbildung 39 zeigt den Schaltplan des Pidgets. Die GPIO Pins 3, 5, 7, 8, 10, 11, und 12 werden als Ausgangspins betrieben. Wird einer dieser Pins auf 3,3V geschaltet, so leuchtet die dazugehörige Leuchtdiode. Die Vorwiderstände  $R_1 - R_7$  ( $240\Omega$ ) begrenzen den Strom durch die LED auf ca. 6mA. Dieser Strom kann mit dem Ohmschen Gesetz wie folgt berechnet werden:

$$I = \frac{U_R}{R_V} = \frac{U_{GPIO} - U_{LED}}{R_V} = \frac{3,3 - 1,9}{240} = 0,0058A \cong 6mA$$

$R_V$  ist der Vorwiderstand,  $U_R$  der Spannungsabfall am Vorwiderstand,  $U_{GPIO}$  die Spannung am Pin (also 3,3V) und  $U_{LED}$  der Spannungsabfall an der Leuchtdiode (dieser ist je nach Diodentyp unterschiedlich, für die verwendete rote LED ist  $U_{LED} = 1,9V$ ).

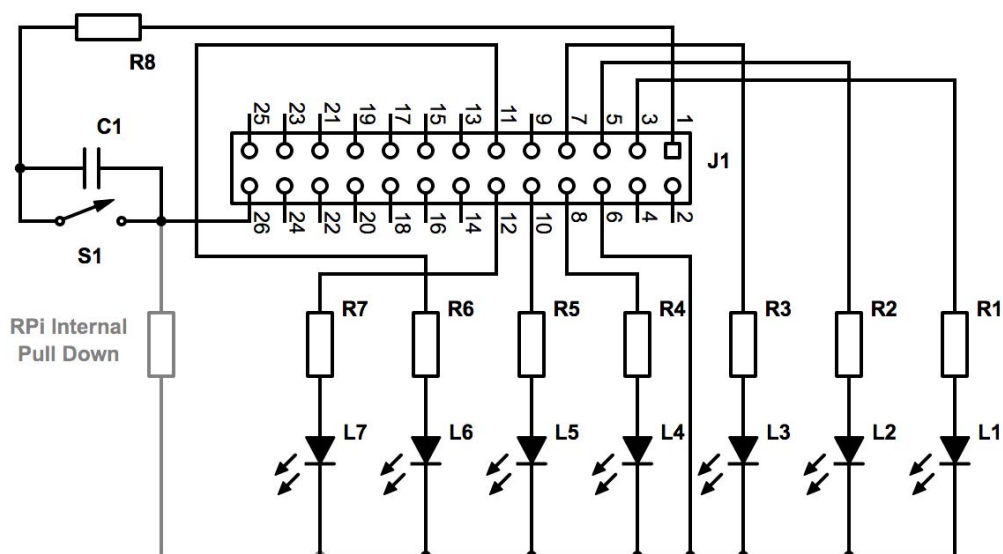


Abbildung 39: Schaltplan Dice

Pin 26 wird als Eingangspin betrieben, über ihn kann der Zustand des Tasters  $S_1$  abgefragt werden. Der Serienwiderstand  $R_8$  schützt Pin 26 vor zu hohen Strömen. Durch Drücken des

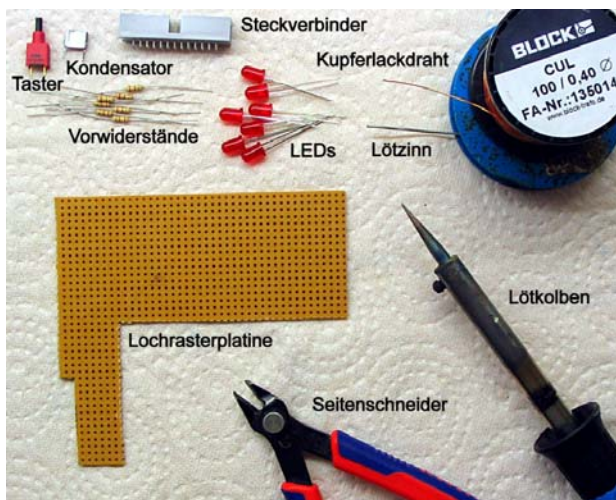
Taster  $S_1$  nimmt Pin 26 den Zustand „high“ ein. Damit der Pin nach Loslassen des Tasters wieder den Zustand „low“ einnimmt, muss beim GPIO Pin 26 der interne Pull-Down Widerstand aktiviert werden (vgl. Kapitel 6.1).

Der Kondensator  $C_1$  hat die Funktion, den Taster zu entprellen. Prellen ist ein Vorgang, der beim Schließen und Öffnen von mechanischen Kontakten passiert. Dabei schalten diese Kontakte schnell mehrfach aus und ein und verursachen so im auslesenden Computerprogramm eine Vielzahl an Schaltimpulsen. Das Prellen kann softwaretechnisch unterdrückt werden. Dies würde für die Schüler eine zusätzliche Schwierigkeit darstellen. Deshalb wird bei den Pidgets versucht, Schalter und Taster hardwaretechnisch zu entprellen.

### 5.4.2 Benötigte Bauteile und Werkzeuge

In

Abbildung 40 sind die benötigten Bauteile und das erforderliche Werkzeug für den Aufbau des Pidgets abgebildet. Zum Prüfen der Schaltung ist ein Multimeter erforderlich. Ist zusätzlich ein Labornetzteil vorhanden, so kann damit die Funktion der Schaltung getestet werden.



#### Werkzeuge

- LötKolben mit Elektronikspitze
- Seitenschneider
- Kupferlackdraht
- Lötzinn

#### Messgeräte

- Multimeter

#### Optional

- Labornetzteil

Abbildung 40: Bauteile und Werkzeuge


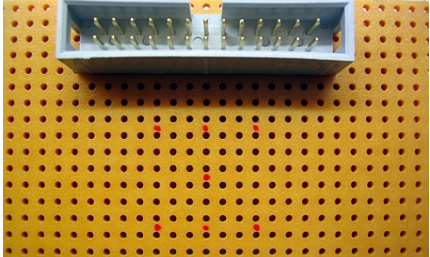
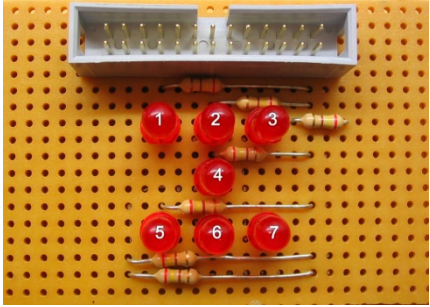
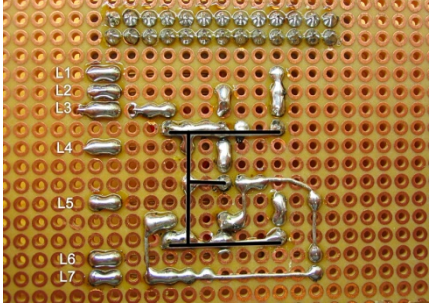
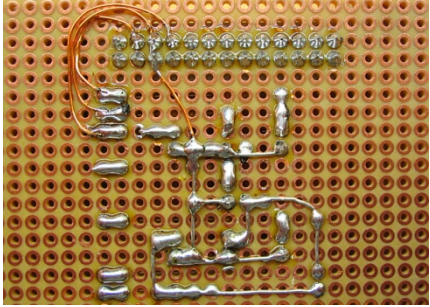
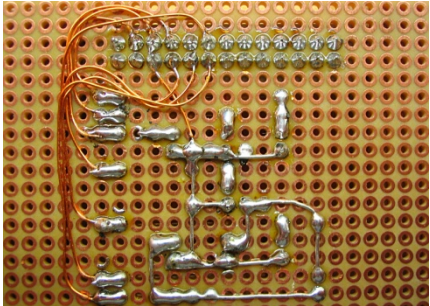
Bauteil	Wert/Bezeichnung	BestellNr.
$R_1$ - $R_7$	$240\Omega$ , 0,25W	7077616
$R_8$	$1800\Omega$ , 0,25W	7077688
$D_1$ - $D_7$	LED, rot, 5mm	2285988
$S_1$	Taster	7023505
$C_1$	22nF, 100V	3121447
$J_1$	Steckverbinder	6741249

Tabelle 28: Stückliste Dice

### 5.4.3 Arbeitsschritte

Tabelle 29 zeigt die Arbeitsschritte, die für den Aufbau des Dice Pidgets notwendig sind.

	<p>Zuerst den 26-poligen Steckverbinder in die Platine einsetzen (so, dass die Ausnehmung im Gehäuse zum Platinenrand zeigt) und verlöten. Dabei ist darauf zu achten, dass der Steckverbinder ganzflächig auf der Platine aufsitzt.</p>
--	--

	<p>Die Lötstellen mit dem Multimeter auf Kurzschlüsse zwischen den Pins kontrollieren!</p>
	<p>Die Positionen der LEDs mit einem Stift markieren. LEDs so einsetzen, dass der längere Anschluss der LED (Anode, Plus-Pol) jeweils durch das Loch oberhalb des Markierungspunktes gesteckt wird. LEDs mit der Platine verlöten.</p>
	<p>Nach den LEDs die Vorwiderstände (<math>240\Omega</math>) einsetzen. Ein Anschluss eines Vorwiderstands wird mit dem Plus-Pol einer LED verlötet, der andere an den (von oben betrachtet) rechten Rand der Schaltung geführt. Die Minus-Pole der LEDs werden untereinander verbunden.</p>
	<p>Die Abbildung zeigt die Rückseite der Platine. Auf der linken Seite sind die Anschlüsse der LEDs 1-7 (samt Vorwiderstand) zu sehen, die Verbindungen der Minus-Pole der LEDs wurden in der Abbildung schwarz gekennzeichnet.</p>
	<p>Mit Kupferlackdraht<sup>*)</sup> die Verbindungen zwischen LED-Anschlüssen und Steckverbinder laut Schaltplan und entsprechend der Pin-Belegung des GPIO Ports (vgl. Abbildung 20) herstellen. In der Abbildung sind die Verbindungen von <math>L_1 \leftrightarrow \text{Pin}3</math>, <math>L_2 \leftrightarrow \text{Pin}5</math>, <math>L_3 \leftrightarrow \text{Pin}7</math> sowie Masse <math>\leftrightarrow \text{Pin}6</math> schon hergestellt. Die restlichen Verbindungen fehlen noch.</p> <p><sup>*)</sup> Kupferlackdraht ist ein mit einer isolierenden Lackschicht überzogener Kupferdraht. Beim Löten verdampft der Lack an der Lötstelle, der restliche Draht bleibt isoliert.</p>
	<p>Restliche Verbindungen herstellen:</p> <ul style="list-style-type: none"> <li>• <math>L_4 \leftrightarrow \text{Pin}8</math></li> <li>• <math>L_5 \leftrightarrow \text{Pin}10</math></li> <li>• <math>L_6 \leftrightarrow \text{Pin}11</math></li> <li>• <math>L_7 \leftrightarrow \text{Pin}12</math></li> </ul>

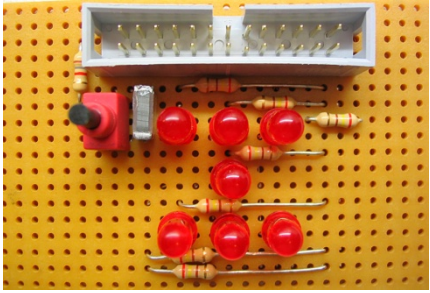
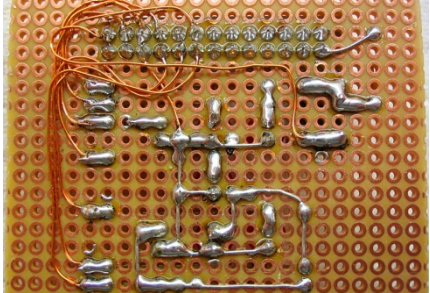
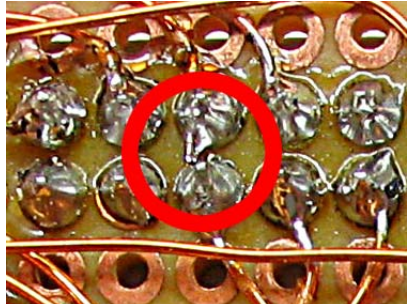
	<p>Taster, Kondensator und Serienwiderstand (1800Ω) einsetzen</p>
	<p>Taster, Kondensator und Serienwiderstand mit der Platine verlöten und mit dem Steckverbinder verbinden: Serienwiderstand <math>R_8 \leftrightarrow</math> Pin12, Taster <math>\leftrightarrow</math> Pin26.</p>
	<p>Damit ist das Pidget fertig gebaut. Was noch fehlt ist die Schaltung auf Verdrahtungsfehler und Kurzschlüsse zu kontrollieren. Dieser Schritt ist besonders wichtig – da Kurzschlüsse (wie im Bild markiert: Verbindung zwischen Pin5 und Pin6) zu dauerhaften Schäden am Raspberry Pi und/oder am Pidget führen können! Es hat sich bewährt, jeden Pin des Steckverbinders mit dem Durchgangsprüfer des Multimeters auf Kurzschlüsse zu seinen Nachbarpins zu prüfen.</p>

Tabelle 29: Arbeitsschritte Pidgetaufbau

#### 5.4.4 Verbindungskabel

Das Verbindungskabel zwischen Raspberry Pi und dem Pidget lässt sich in wenigen Arbeitsschritten anfertigen, wie Tabelle 30 zeigt.

	<p>Für die Herstellung des Verbindungskabels werden</p> <ul style="list-style-type: none"> <li>• 2 Steckbuchsen, 26-polig</li> <li>• ein Stück Flachbandkabel in gewünschter Länge benötigt.</li> </ul>
	<p>Beim Kabel markiert die rote Ader den 1. Pin, bei der Steckbuchse das kleine Dreieck.</p>
	<p>Das Kabel so durch die Steckbuchse stecken, dass sich die rote Ader bei Pin1 befindet, das Kabelende mit der unteren Seite der Steckbuchse plan abschließt und die „Nase“ der Steckbuchse nach oben schaut. Steckbuchse und Kabel in einen Schraubstock geben und die Backen des Schraubstocks schließen.</p>


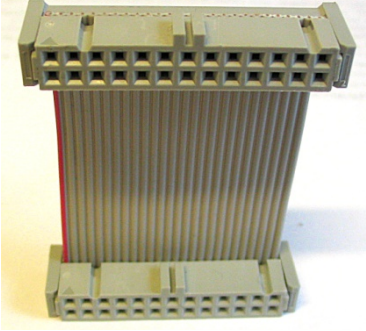
	<p>Steckbuchse aus dem Schraubstock ausspannen und die zweite Kabelbuchse am anderen Ende des Kabels montieren. Wichtig dabei ist, dass die Nase der zweiten Kabelbuchse in die gleiche Richtung zeigt, wie die der ersten.</p>
	<p>So sieht das fertige Verbindungskabel aus. Die so genannten „Nasen“ beider Kabelbuchsen zeigen nach oben, die rote Ader (Pin1 des Flachbandkabels) ist links und das Kabel schließt bündig mit der Oberkante der Kabelbuchse ab.</p>

Tabelle 30: Arbeitsschritte Kabelherstellung

## 6 Pidget-Module

In diesem Kapitel befinden sich für jedes Pidget die Beschreibung und der Quellcode für ein Python-Modul, das die Schüler in ihre Programme importieren können. Mit Hilfe dieser Module können die Pidget-Funktionen über Methodenaufrufe genutzt werden. Ein Modul für ein Pidget zu programmieren oder zu erweitern, kann auch als Aufgabenstellung im Informatikunterricht gegeben werden. Dies ist im Sinne der in der Einleitung dargestellten Meinung des Autors, dass ein Pidget für die Schüler keine Black Box darstellen soll, sogar wünschenswert. Für diesen Fall stellen die Listings in diesem Kapitel die Musterlösungen dar.

Es existieren für alle Pidgets auch Testprogramme, in denen die Pidget-Module verwendet werden. Ein solches Testprogramm zeigt Listing 3. Sämtliche Module und Testprogramme können von Google Code heruntergeladen werden. [30]

Jedes Pidget-Modul verfügt über eine Methode `init()`, die die verwendeten Pins des GPIO Ports als Aus- bzw. Eingang initialisiert und über eine Methode `description()`, welche eine Liste der vorhandenen Methoden am Bildschirm ausgibt.

Ein Anwendungsbeispiel, wie das Modul `Dice.py` verwendet werden kann, zeigt Listing 3:

```
#!/usr/bin/env python
import time
import random
import Dice as dice

dice.init()
dice.test()
dice.description()

# counting from 1 to 6
for i in range(1,7):
    dice.set_dice(i)
    time.sleep(0.5)

dice.set_dice(0)

# generating random numbers by pressing switch
while True:
    if dice.get_switch():
        dice.set_dice(random.randrange(1,7))
        time.sleep(0.05)
```

Listing 3: Nutzung von `Dice.py`

### 6.1 Dice

Dieses Modul verwendet für die Zuordnung der gewürfelten Zahlen zu den LEDs (pin-mapping) den Datentyp Dictionary. Jeder Eintrag besteht aus einem Schlüssel (Zahlen [1,6]) und einem Wert (jeweils eine Liste der GPIO Pins, die eingeschaltet werden müssen). Werden Schlüssel übergeben für die kein Eintrag existiert, so werden alle LEDs ausgeschaltet.

#### Quellcode

```
import RPi.GPIO as GPIO
import time

# use physical pin numbers
GPIO.setmode(GPIO.BOARD)
# switch off GPIO warnings
```

```

GPIO.setwarnings(False)

output_pin = [3,5,7,8,10,11,12]
input_pin = 26
pin_mapping = {1:[8],2:[3,12],3:[3,8,12],4:[3,7,10,12],5:[3,7,8,10,12],6:[3,5,7,10,11,12]}

def init():
    # setup output pins, switch pins off
    for i in output_pin:
        GPIO.setup(i,GPIO.OUT,initial=0)

    # setup input pin; enable internal pull down resistor
    GPIO.setup(input_pin,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)

def description():
    print ('dice pidget methods:')
    print ('  init():          initializes input and output pins')
    print ('  set_dice([0,6]): sets LEDs to given number')
    print ('  get_switch():     returns state of switch')
    print ('  test():          flashes all 7 LEDs for 1 second')

def set_dice(number):
    for i in output_pin:
        GPIO.output(i,0)

    if number in pin_mapping:
        for i in pin_mapping[number]:
            GPIO.output(i,1)

def get_switch():
    return GPIO.input(26)

def test():
    for i in output_pin:
        GPIO.output(i,1)

    time.sleep(1)

    for i in output_pin:
        GPIO.output(i,0)

```

Listing 4: Dice.py

## 6.2 LED Strip

Dieses Modul schaltet die LEDs [0,7] des Pidgets ein oder aus. Die zu den LEDs zugeordneten GPIO Pins sind in einem Array abgelegt, der Zugriff erfolgt über den Arrayindex.

### Hinweise

- In Python ist es möglich über Index -1,-2... auf das letzte, vorletzte... Element eines Arrays zuzugreifen.
- Um einen GPIO Pin einzuschalten lautet der Befehl `GPIO.output(i,x)`, wobei x jede Zahl außer 0 sein darf.
- Der Status eines Output Pins kann abgefragt werden, indem dieser Pin als Input angesprochen wird (z.B.: `print(GPIO.input(PinNr))`)
- Der Status eines Output Pins kann mit folgender Anweisung umgeschaltet werden: `GPIO.output(PinNr, not GPIO.input(PinNr))`



Die Methode `set_binary(number)` soll die übergebene Zahl als Binärzahl am LED Strip anzeigen. Von dieser Methode ist nur ein Gerüst angegeben. Sie soll von den Schülern selbst implementiert werden (vgl. Kapitel 7.2).

## Quellcode

```
import RPi.GPIO as GPIO
import time

# use physical pin numbers
GPIO.setmode(GPIO.BOARD)

# switch off GPIO warnings
GPIO.setwarnings(False)

output_pin = [3,5,7,8,10,11,12,13]

def init():
    # setup output pins, switch pins off
    for i in output_pin:
        GPIO.setup(i,GPIO.OUT,initial=0)

def description():
    print('LED strip pidget methods:')
    print('  init():                initializes output pins')
    print('  set_led([0,7],[0,1]): sets LED #0-7 to off or on')
    print('  set_binary([0,255]): LEDs show binary representation of given number')
    print('  get_led_states():      returns state of LEDs as integer')
    print('  test():                flashes all 8 LEDs for 1 second')

def set_led(number,status):
    try:
        GPIO.output(output_pin[int(number)],int(status))
    except ValueError:
        print ('Error: number/status is not an integer')
    except IndexError:
        print ('Error: number not in range [0,7]')

def set_binary(number):
    try:
        number=int(number)

        if (number>=0 and number<256):
            # to be done by students:
            # LEDs should show a binary representation of number
            print('Not implemented, yet!')
        else:
            print('Error: number not in range [0,255]')
            # switch off all LEDs
            set_binary(0)
    except ValueError:
        print ('Error: number not an integer')

def get_led_states():
    state = 0

    for i in range (0,8):
        state = state + GPIO.input(output_pin[i]) * 2**i

    return state

def test():
    for i in output_pin:
        GPIO.output(i,1)
```

```
time.sleep(1)

for i in output_pin:
    GPIO.output(i,0)
```

Listing 5: LED\_Strip.py

### 6.3 7-Segment Display

Das 7-Segment Display besteht aus vier Ziffern und einem Taster. Die Ziffern können auf drei verschiedene Arten angesprochen werden:

- `set_display(number)` stellt eine Zahl zwischen 0 und 9999 (mit führenden Nullen) dar.
- `set_digit(digit,number)` stellt an der Position `digit` [0,3] die Ziffer `number` [0,9] dar. Die Positionen (Stellen) sind von rechts nach links nummeriert. Bei Zahlen größer als 9 wird die Stelle ausgeschaltet.
- `set_double(double,number)` stellt im Ziffernpaar `double` [0,1] eine zweistellige Zahl `number` [0-99] dar. Die Ziffernpaare sind von rechts nach links nummeriert.

#### Quellcode

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import time

# use physical pin numbers
GPIO.setmode(GPIO.BOARD)

# switch off GPIO warnings
GPIO.setwarnings(False)

output_pin = [3,5,7,8,10,11,12,13,15,16,18,19,21,22,23,24]
input_pin = 26

def init():
    # setup output pins, switch pins off
    for i in output_pin:
        GPIO.setup(i,GPIO.OUT,initial=0)

    # setup input pin; enable internal pull down resistor
    GPIO.setup(input_pin,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)

def description():
    print ('7-Segment Display pidget methods:')
    print ('  init():                initializes input and output pins')
    print ('  set_display(number):      sets Display to given number')
    print ('  set_digit(digit,number):  sets digit to given number')
    print ('  set_double_digit(double,number): sets two digits to number')
    print ('  get_switch():            returns state of switch')

def set_digit(digit,number):
    try:
        number=int(number)
        digit=int(digit)

        if (digit<0 or digit>3):
            print ('digit not in range [0,3]')
        else:
            if (number>9 or number<0):
                # set digit blank
                number=10
```

```

        for i in range (0,4):
            GPIO.output(output_pin[i+4*digit],(number & (1 << i)))
    except ValueError:
        print ('Error: digit/number is not an integer')

def set_display(number):
    try:
        number=int(number)
        if (number<0 or number>9999):
            print ('number not in range [0,9999]')
        else:
            for i in range (0,4):
                set_digit(i,int((number/10**i)%10))
    except ValueError:
        print ('Error: number is not an integer')

def set_double_digit(double,number):
    try:
        number=int(number)
        double=int(double)

        if (number<0 or number>99):
            print ('number not in range [0,99]')
        elif (double<0 or double>1):
            print ('double not in range [0,1]')
        else:
            for i in range (0,2):
                set_digit(i+double*2,int((number/10**i)%10))
    except ValueError:
        print ('Error: double/number is not an integer')

def test():
    number = 1111

    for i in range (0,10):
        set_display(i*number)
        time.sleep(0.5)

    set_display(0)

def get_switch():
    return GPIO.input(26)

```

Listing 6: Seven\_Segment\_Display.py

## 6.4 Optical Transmitter

Dieses Modul schaltet den Laser und die LED des Pidgets ein oder aus. Es stehen folgende Methoden zur Verfügung:

- `set_laser(state)` schaltet den Laser ein oder aus. Um den Laser einzuschalten, darf `state` jeden ganzzahligen Wert außer 0 annehmen.
- `set_laserpulse(length)` schaltet den Laser für die Zeit `length` ein, wobei `length` in Sekunden anzugeben ist. Während dieser Zeit ist das Python-Programm im Warte-Modus.
- `get_state()` liefert den Zustand des Lasers (ein oder aus) als Boolesche Variable zurück.

## Quellcode

```

import RPi.GPIO as GPIO
import time

# use physical pin numbers
GPIO.setmode(GPIO.BOARD)

# switch off GPIO warnings
GPIO.setwarnings(False)

def init():
    # setup output pin, switch pin off
    GPIO.setup(3,GPIO.OUT,initial=0)

def description():
    print('Optical Transmitter pidget methods:')
    print('  init():                initializes output pin')
    print('  set_laser([0,1]):        sets laser to on or off')
    print('  set_laserpulse(length): switches laser on for length seconds')
    print('  get_state():             returns state of laser as boolean')
    print('  test():                  flashes laser for 1 second')

def set_laser(state):
    try:
        GPIO.output(3,int(state))
    except ValueError:
        print ('Error: state is not an integer')

def set_laserpulse(length):
    try:
        length=float(length)
        GPIO.output(3,1)
        time.sleep(length)
        GPIO.output(3,0)
    except ValueError:
        print ('Error: length is not a number')

def get_state():
    return GPIO.input(3)

def test():
    GPIO.output(3,1)
    time.sleep(1)
    GPIO.output(3,0)

```

Listing 7: Optical\_Transmitter.py

## 6.5 Optical Receiver

Dieses Modul hat keine test() Methode – weil diese Methode ident wäre mit get\_state() (liefert den Zustand des Receivers als Boolesche Variable zurück).

## Quellcode

```

import RPi.GPIO as GPIO

# use physical pin numbers
GPIO.setmode(GPIO.BOARD)

# switch off GPIO warnings
GPIO.setwarnings(False)

input_pin = 26

```

```
def init():
    # setup input pin, enable pull down resistor
    GPIO.setup(input_pin,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)

def description():
    print('Optical Receiver pidget methods:')
    print('  init():      initializes input pin')
    print('  get_state(): returns state of receiver')

def get_state():
    return GPIO.input(26)
```

Listing 8: Optical\_Receiver.py

## 7 Programmieraufgaben

Dieses Kapitel beinhaltet eine Sammlung von Programmieraufgaben für den Raspberry Pi und die realisierten Pidgets. Es wird für die Aufgaben vorausgesetzt, dass die Schüler bereits grundlegende Programmierkenntnisse erlangt haben (vgl. Kapitel 1.3). Darüber hinaus wird davon ausgegangen, dass den Schülern im Zuge des Informatikunterrichts gezeigt wurde, wie ein Pidget mit dem Raspberry Pi verbunden wird und wie die Module aus dem Kapitel 6 verwendet werden können. In diesem Zusammenhang wird noch einmal auf die Methode `description()` hingewiesen, die für jedes Pidget existiert und am Bildschirm eine Liste der Methoden für ein Pidget ausgibt. Der Lehrplan für den Wahlpflichtgegenstand (WPG) Informatik ist ein Rahmenlehrplan ohne konkrete Verknüpfung der Lehrinhalte mit der Schulstufe. Deshalb können auch die einzelnen Aufgaben nicht einer bestimmten Schulstufe zugewiesen werden.

Die Aufgaben sind nach folgendem Schema aufgebaut:

- In einer **Lehrerinformation** werden didaktische und pädagogische Aspekte erläutert. Das Kompetenzmodell, auf das Bezug genommen wird, entstammt dem „Leitfaden zur kompetenzorientierten Reifeprüfung an AHS“. [31]
- Die Aufgabenstellung ist in Form eines **Arbeitsblattes** gestaltet. Bonusaufgaben sind Angebote an leistungsstärkere Schüler. Die restlichen Aufgaben sind so gestellt, dass sie von allen Schülern gelöst werden können.
- Eine **Musterlösung** wird in Form des Quellcodes angegeben und
- **Antworten** auf Fragen im Arbeitsblatt in Form einer Tabelle.

Wie schon in Kapitel 1.4 erwähnt, sind die Arbeitsblätter so gestaltet, dass die Schüler bei der Bearbeitung der Aufgaben große Unterstützung erfahren. Auch die Unterrichtsformen sind so gewählt, dass es immer wieder Möglichkeiten gibt, die Schüler auf einen Wissens- und Implementierungsstand zu bringen. Bei leistungsstärkeren (höheren) Klassen kann die Unterstützung durch die Arbeitsblätter oder die Lehrperson natürlich zurückgenommen und von den Schülern mehr selbständiges Arbeiten gefordert werden.

Die Musterlösungen können von Google Code heruntergeladen werden. [30]

Damit die Schüler die Aufgabenstellung besser verstehen können und auch um sie für die Aufgabenstellung zu motivieren, wird empfohlen den Schülern zu Beginn einer jeden Aufgabe Pidget und Musterlösung in Aktion zu zeigen und ggf. ausprobieren zu lassen.

### 7.1 Lauflicht

#### 7.1.1 Lehrerinformation

Verwendetes Pidget	LED Strip
Alltagsbezug	Lauflichter begegnen den Schülern regelmäßig im Alltag – von optischen Effekten in Lokalen und Geschäften bis zu sicherheitstechnischen Einrichtungen im Straßenverkehr. Das Lauflicht im Kühlergrill des K.I.T.T. (aus der US-Fernsehserie „Knight Rider“) ist wahrscheinlich manchen Schülern auch noch bekannt.
Motivation für die Aufgabe	Diese Aufgabe ist ein guter Einstieg in die Verwendung von Pidgets. Sie ist außerdem dazu geeignet, die Inhalte des vorangegangenen Programmierunterrichts zu wiederholen und zu festigen. Durch die Bonusaufgaben wird eine innere Differenzierung ermöglicht. Die

	Teilaufgabe 4 enthält eine fächerübergreifende Komponente (Physik: Schwingungen). Das fördert die Fähigkeit der Schüler, Problemstellungen aus verschiedenen Disziplinen in einem Computerprogramm zu modellieren.
Lernziele	<ul style="list-style-type: none"> <li>• Die Schüler sind in der Lage Grundkenntnisse der Programmierung anwenden zu können (siehe „Nötiges Vorwissen“)</li> <li>• Die Schüler sollen einfache verbale Aufgabenstellungen (<i>Solange der Benutzer ungültige Werte eingibt, soll er zu einer erneuten Eingabe aufgefordert werden</i>) in die Sprache der Programmierung (eine <i>while</i>-Schleife) transferieren können.</li> <li>• Die Schüler sollen sinnvolle Testfälle konstruieren und damit Tests durchführen können.</li> </ul>
Schulstufe	WPG (Wahlpflichtgegenstand, 6.-8. Klasse)
Nötiges Vorwissen	<p>Schleifen (for, while)                  Typumwandlungen (string, int, float)                  String Konkatination                  Wertevergleich (&lt;, &gt;, =, !=)                  Methoden und Methodenaufrufe                  Benutzereingaben von Tastatur                  Ausnahmebehandlung (try, except)                  Einfache arithmetische Berechnungen (+, -, *, /, %)                  Modul time (sleep)                  Bonusaufgaben:                  Modul time (localtime)                  Werte runden (round)</p>
Bezug Kompetenzmodell	<p>Inhaltsdimension „Praktische Informatik“</p> <ul style="list-style-type: none"> <li>• Algorithmen, Datenstrukturen und Programmierung</li> </ul> <p>Handlungsdimension „Anwenden und Gestalten“</p> <ul style="list-style-type: none"> <li>• Aufgaben mit Mitteln der Informatik modellieren</li> <li>• Algorithmen entwerfen, implementieren und testen</li> </ul> <p>Reflektieren und Bewerten</p> <ul style="list-style-type: none"> <li>• Gezielt nach Programmfehlern suchen und diese korrigieren</li> </ul>
Unterrichtsmethode	Einzelarbeit, um das selbständige Arbeiten zu fördern und dem Schüler ein individuelles Arbeitstempo zu ermöglichen. Die Lehrperson kann bei Problemen Schüler individuell unterstützen.
Dauer	2 Doppelstunden
Anmerkungen	Die Teilaufgaben sind so gewählt, dass das Programm in kleinen Schritten wächst und der bereits erstellte Code weiterverwendet werden kann. Die Gesamtaufgabe hat spielerischen Charakter. Es ist zu erwarten, dass die Schüler mit sehr kurzen Zykluszeiten experimentieren – dies führt dazu, dass für das menschliche Auge alle LEDs mit verminderter Helligkeit leuchten. Hier kann die Lehrperson auf die Methode der Pulsweitenmodulation zur Helligkeitsregelung hinweisen [32] und ggf. Zusatzaufgaben stellen.

Tabelle 31: Lauflicht Lehrerinformation





<p>Aufgabe 4</p>	<p>Erweitere das Programm, sodass der Benutzer die „Geschwindigkeit“ des Lauflichts über die Tastatur eingeben kann. Die Geschwindigkeit ist, die Anzahl der vollständigen Zyklen pro Sekunde. Der Dialog auf der Konsole soll so aussehen:</p> <pre>Please enter number of cycles [1-100]: 5 Please enter speed [cycles per second]: 0.5 Doing 5 cycles @ 0.5 cycle(s) per second. Done.</pre> <p>Solange der Benutzer ungültige Werte eingibt, soll er zu einer erneuten Eingabe aufgefordert werden:</p> <pre>Please enter speed [cycles per second]: -3 Input is not valid, please reenter speed:</pre> <p><b>Implementierungshinweise</b> Die Benutzereingabe soll in der Variable <code>speed</code> gespeichert werden. Die Eingabe des Benutzers soll (ähnlich wie in Aufgabe 3) in einer Methode (<code>is_valid_speed(num)</code>) überprüft werden.</p> <p>Gib hier einige <b>gültige</b> Eingaben für <code>speed</code> an:</p> <p>Gib hier einige <b>ungültige</b> Eingaben für <code>speed</code> an:</p> <p>Teste dein Programm mit gültigen <b>und</b> ungültigen Werten.</p> <p>Aus dem Wert der Variable <code>speed</code> soll die Leuchtdauer (<code>delay</code>) pro LED errechnet werden. Die Dauer (<math>T</math>) für einen Zyklus ist der Kehrwert der Zyklen pro Sekunde (<math>f</math>). Erinnere dich an den Physikunterricht und den Zusammenhang zwischen Frequenz und Periodendauer: <math>T = 1/f</math>. Die Leuchtdauer (<code>delay</code>) einer LED errechnet sich zu <math>delay = T/x = 1/(f \cdot x)</math>, wobei <math>x</math> die Anzahl der Schritte für einen kompletten Zyklus ist. Überlege, nach wie vielen Schritten, der Zyklus von vorne beginnt.</p>
<p>Bonus 1</p>	<p>Implementiere eine Zeitmessung zum Überprüfen der Ergebnisse von Aufgabe 4. Die Messung soll die Dauer liefern, die das Lauflicht für die Ausführung aller Zyklen benötigt und diesen Wert am Ende des Programms anzeigen. Der Wert soll in Sekunden und auf zwei Kommastellen gerundet angezeigt werden:</p> <pre>Please enter number of cycles [1-100]: 7 Please enter speed [cycles per second]: 0.33 Doing 5 cycles @ 0.5 cycle(s) per second. Done. Duration was 21.48 seconds.</pre> <p><b>Implementierungshinweise:</b> <code>time.time()</code> liefert dir die aktuelle Systemzeit. Speichere diesen Wert vor und nach dem Programmteil, der das Lauflicht ausmacht. Für die Ausgabe kannst du <code>round(num, x)</code> verwenden, das die Zahl <code>num</code> auf <code>x</code> Kommastellen rundet. Überlege dir Testparameter für Anzahl der Zyklen und Zyklen pro Sekunde um das Ergebnis einfach überprüfen zu können. Zyklen: _____ Zyklen pro Sekunde: _____ Erwartetes Ergebnis: _____ s Überlege und notiere, warum das Ergebnis vom erwarteten Wert abweicht:</p>
<p>Bonus 2</p>	<p>Ändere das Programm, sodass das Lauflicht wie in diesem Video (<a href="http://www.myvideo.at/watch/4890993">http://www.myvideo.at/watch/4890993</a>) hin und her läuft. Achte darauf, dass die LEDs am Rand (LED0 und LED7) nicht länger leuchten als die anderen LEDs!</p>

### 7.1.3 Musterlösung

```
#!/usr/bin/env python
import time
import LED_Strip as ledstrip

ledstrip.init()

def is_valid_cycles(num):
    try:
        return (int(num)>0 and int(num)<101)
    except ValueError:
        return False

def is_valid_speed(num):
    try:
        return float(num)>0
    except ValueError:
        return False

cycles = input('Please enter number of cycles [1-100]: ')

while not is_valid_cycles(cycles):
    cycles = input('Input is not valid, please reenter number of cycles: ')

cycles = int(cycles)

speed = input('\nPlease enter speed [cycles per second]: ')

while not is_valid_speed(speed):
    speed = input('Input is not valid, please reenter speed: ')

# 1 cycle (back and forth) is 14 steps
delay = 1/(14*float(speed))

print ('\nDoing ' + str(cycles) + ' cycles @ ' + speed + ' cycle(s) per second.')

start = time.time()

for k in range (0, cycles):
    # chasing LEDs 0 to 7
    for i in range (0,8):
        ledstrip.set_led(i,1)
        time.sleep(delay)
        ledstrip.set_led(i,0)

    # chasing LEDs 6 to 1
    for i in range (1,7):
        ledstrip.set_led(7-i,1)
        time.sleep(delay)
        ledstrip.set_led(7-i,0)
ledstrip.set_led(0,1)
time.sleep(delay)

stop = time.time()

# switching off all LEDs
for i in range (0,8):
    ledstrip.set_led(i,0)

print ('\nDone. Duration was ' + str(round((stop-start),2)) + ' seconds.')
```

Listing 9: Light\_Chaser.py

### 7.1.4 Musterantworten

Aufgabe	Antwort
Aufgabe 1 – Datentyp	Typ für <code>delay</code> ist <code>float</code> , weil die Leuchtdauer auch ein Bruchteil einer Sekunde sein kann.
Aufgabe 2 – Datentyp	Typ für <code>cycles</code> ist <code>integer</code> , weil die Anzahl der Zyklen immer eine ganze Zahl ist.
Aufgabe 3 – gültige und ungültige Eingaben für <code>cycles</code>	Gültig: ganze positive Zahlen zw. 1 und 100 Ungültig: 1,2; 1.2; a; -3
Aufgabe 4 – gültige und ungültige Eingaben für <code>speed</code>	Gültig: positive Integer- oder Gleitkommazahlen Ungültig: 1,2; -1.2; a
Bonus 1	1 Zyklus pro Sekunde, 10 Zyklen → Erwartetes Ergebnis: 10s Ergebnis weicht ab, weil die restliche Programmausführung Zeit benötigt → Dauer ist länger als erwartet.

Tabelle 32: Lauflicht Musterantworten

## 7.2 Binärzahlendarstellung

### 7.2.1 Lehrerinformation

Verwendetes Pidget	LED Strip																
Alltagsbezug	Bei dieser Aufgabe kann der Alltagsbezug durch eine binäre Uhr hergestellt werden. Eine solche Uhr stellt die Uhrzeit meist mittels Leuchtdioden in binärer Form da. Das LED Strip Pidget kann als „Minutenzeiger“ einer solchen Uhr aufgefasst werden.																
Motivation für die Aufgabe	Motivation für diese Aufgabe ist, mit den Schülern die binäre Darstellung von Zahlen und die Umrechnung zwischen den Zahlensystemen zu wiederholen und zu festigen. Durch das zusätzlich zu erstellende Testprogramm sehen die Schüler das binäre Zählen am Pidget und erhalten so eine bessere Vorstellung davon, wie Zahlen im Speicher eines Rechners dargestellt werden. Die Lösung der Aufgabe wird für Aufgabe 7.3 benötigt																
Lernziele	<ul style="list-style-type: none"> <li>• Grundkenntnisse der Programmierung sollen angewendet und gefestigt werden</li> <li>• Schüler sollen ein Struktogramm in ein Programm transferieren können.</li> <li>• Testfälle sollen überlegt und durchgeführt werden.</li> </ul>																
Schulstufe	WPG (Wahlpflichtgegenstand, 6.-8. Klasse)																
Nötiges Vorwissen	Schleifen (for, while) Typumwandlungen (string, int, float) Wertevergleich (<, >, ==, !=) Ausnahmebehandlung (try, except) Einfache arithmetische Berechnungen (+, -, *, /, %)																
Bezug Kompetenzmodell	Inhaltsdimension „Praktische Informatik“ <ul style="list-style-type: none"> <li>• Algorithmen, Datenstrukturen und Programmierung</li> </ul> Handlungsdimension „Anwenden und Gestalten“ <ul style="list-style-type: none"> <li>• Aufgaben mit Mitteln der Informatik modellieren</li> <li>• Algorithmen entwerfen, implementieren und testen</li> </ul>																
Unterrichtsmethode	Einzelarbeit																
Dauer	1 Doppelstunde																
Anmerkungen	<p>Die vorletzte Anweisung im Struktogramm <code>number = int(number/2)</code> sollte noch erläutert werden. Die Funktion des „/“ Operators ist seit Python 3.x eine Gleitkommadivision. Für die Ganzzahldivision wurde der „//“ Operator eingeführt. Aus Gründen der Rückwärtskompatibilität werden in dieser Arbeit alle Ganzzahldivisionen als explizierter Typecast <code>int(a/b)</code> codiert. Tabelle 33 zeigt die unterschiedlichen Wirkungsweisen.</p> <table border="1" data-bbox="593 1780 1409 2011"> <thead> <tr> <th>Python 2.x</th> <th>Python 3.x</th> </tr> </thead> <tbody> <tr> <td><code>&gt;&gt;&gt; 10.0/3</code></td> <td><code>&gt;&gt;&gt; 10 / 3</code></td> </tr> <tr> <td><code>3.3333333333333335</code></td> <td><code>3.3333333333333335</code></td> </tr> <tr> <td><code>&gt;&gt;&gt; 10/3</code></td> <td><code>&gt;&gt;&gt; 10 // 3</code></td> </tr> <tr> <td><code>3</code></td> <td><code>3</code></td> </tr> <tr> <td><code>&gt;&gt;&gt; int(10/3)</code></td> <td><code>&gt;&gt;&gt; int(10/3)</code></td> </tr> <tr> <td><code>3</code></td> <td><code>3</code></td> </tr> <tr> <td><code>&gt;&gt;&gt;</code></td> <td><code>&gt;&gt;&gt;</code></td> </tr> </tbody> </table>	Python 2.x	Python 3.x	<code>&gt;&gt;&gt; 10.0/3</code>	<code>&gt;&gt;&gt; 10 / 3</code>	<code>3.3333333333333335</code>	<code>3.3333333333333335</code>	<code>&gt;&gt;&gt; 10/3</code>	<code>&gt;&gt;&gt; 10 // 3</code>	<code>3</code>	<code>3</code>	<code>&gt;&gt;&gt; int(10/3)</code>	<code>&gt;&gt;&gt; int(10/3)</code>	<code>3</code>	<code>3</code>	<code>&gt;&gt;&gt;</code>	<code>&gt;&gt;&gt;</code>
Python 2.x	Python 3.x																
<code>&gt;&gt;&gt; 10.0/3</code>	<code>&gt;&gt;&gt; 10 / 3</code>																
<code>3.3333333333333335</code>	<code>3.3333333333333335</code>																
<code>&gt;&gt;&gt; 10/3</code>	<code>&gt;&gt;&gt; 10 // 3</code>																
<code>3</code>	<code>3</code>																
<code>&gt;&gt;&gt; int(10/3)</code>	<code>&gt;&gt;&gt; int(10/3)</code>																
<code>3</code>	<code>3</code>																
<code>&gt;&gt;&gt;</code>	<code>&gt;&gt;&gt;</code>																

Tabelle 33: Divisionen in Python 2.x und 3.x

### 7.2.2 Arbeitsblatt

<b>Binärzahlendarstellung</b>	
<p>Erweitere das Modul LED_Strip.py um die Methode set_binary(number), die number als Binärzahl am LED Strip darstellt.</p>	
<p>Aufgabe 1</p>	<p>Öffne das Modul LED_Strip.py und vervollständige die Methode set_binary(number). Ersetze die beiden Kommentarzeilen (,to be done by...‘, ,LEDs should show...‘) und die Bildschirmausgabe (,Not implemented, yet!‘) durch ein Programm, das number als Binärzahl am LED Strip ausgibt. Die folgende Abbildung zeigt ein Struktogramm des Programms.</p> <div style="border: 1px solid black; padding: 10px; margin: 10px auto; width: fit-content;"> <p style="text-align: center;"><b>set_binary(number)</b></p> <pre> i = 0 to 7     set_led(i,0)  i = 0 while number != 0     set_led(i,number mod 2)     number = int(number/2)     i = i+1                     </pre> </div>
<p>Aufgabe 2</p>	<p>Schreibe ein Programm Binary_Displaytest.py, das deine Methode testet. Das Programm soll das Modul LED_Strip.py verwenden und von der kleinsten bis zur größten am LED Strip darstellbaren Zahl alle Zahlen der Reihe nach anzeigen. Jede Zahl soll 0,25s lang angezeigt werden.</p> <p>Überlege und notiere:</p> <p>Was ist die kleinste Zahl, die am LED Strip angezeigt werden kann? _____</p> <p>Was ist die größte Zahl, die am LED Strip angezeigt werden kann? _____</p>
<p>Bonus</p>	<p>Eine sehr elegante Möglichkeit, Dezimalzahlen in Binärzahlen umzuwandeln sind Bitoperationen. Der Code für die Umwandlung einer achtstelligen Zahl sieht so aus:</p> <pre> for i in range (0,8):     set_led(i,(number &amp; (1 &lt;&lt; i)))                     </pre> <p>Arbeite die Seite <a href="http://wiki.python.org/moin/BitwiseOperators">http://wiki.python.org/moin/BitwiseOperators</a> durch und erkläre was die obigen Anweisungen bewirken:</p>

Tabelle 34: Binärzahlendarstellung Lehrerinformation

### 7.2.3 Musterlösungen

Methode set\_binary(number)

```
def set_binary(number):
    try:
        number=int(number)

        if (number>=0 and number<256):
            for i in output_pin:
                GPIO.output(i,0)
                i=0

            while number!=0:
                set_led(i,number%2)
                number = int(number/2)
                i = i+1
            else:
                print('Error: number not in range [0,255]')
                # switch off all LEDs
                set_binary(0)
    except ValueError:
        print ('Error: number not an integer')
```

Listing 10: Methode set\_binary(number)

Programm Binary\_Displaytest.py

```
#!/usr/bin/env python
import LED_Strip as ledstrip
import time

ledstrip.init()

for i in range (0,256):
    ledstrip.set_binary(i)
    time.sleep(0.25)
```

Listing 11: Binary\_DisplayTest.py

### 7.2.4 Musterantworten

Aufgabe	Antwort
Aufgabe 2 – Wertebereich	Kleinste darstellbare Zahl: 0 Größte darstellbare Zahl: 255
Bonus	Die for-Schleife iteriert über die 8 Stellen des LED Strips. Die i-te Stelle wird eingeschaltet, wenn der bitweise Vergleich (&) zwischen der Zahl und $2^i$ nicht null ergibt. $2^i$ wird erzeugt, in dem 1 um i Stellen nach links geschoben wird ( $1 \ll i$ ). Der Vergleich liefert als Ergebnis $2^i$ oder null.

Tabelle 35: Binärzahlendarstellung Musterantworten

## 7.3 Binary Battle

### 7.3.1 Lehrerinformation

Verwendetes Pidget	LED Strip
Alltagsbezug	Computerspiele sind für viel Schüler faszinierend und sind aufgrund der Leistungsfähigkeit heutiger Mobiltelefone im Leben der Schüler „immer mit dabei“.
Motivation für die Aufgabe	<p>Die erste Motivation ist, dass die Schüler mit deutlich weniger als 100 Zeilen Code ein funktionierendes Spiel programmieren können. Somit ist die Erfolgswahrscheinlichkeit hoch und der Aufwand bis zum Erreichen des Ziels überschaubar.</p> <p>Der scheinbar spielerische Charakter der Aufgabenstellung hat in Wirklichkeit den Hintergrund, dass sich die Schüler noch einmal mit den Binärzahlen beschäftigen und am Pidget visualisiert bekommen, wie Zahlen im Computer repräsentiert sind.</p> <p>Die Aufgabe bietet schließlich die Möglichkeit, verschiedenen Unterrichtsmethoden (Einzelarbeit, Teamarbeit, Diskussion der Ergebnisse mit der gesamten Gruppe) einzusetzen. Dadurch können die Schüler die Zusammenarbeit mit anderen üben und die höhere Problemlösungskompetenz eines Teams erleben.</p>
Lernziele	<ul style="list-style-type: none"> <li>• Die Schüler sollen eine Aufgabe analysieren und in Teilaufgaben zerlegen können.</li> <li>• Die Schüler sollen Fachwissen aus Informatik (Darstellung von Zahlen im Speicher) und Mathematik (Potenzrechnung) vernetzen können.</li> <li>• Die Schüler sollen sinnvolle Testfälle konstruieren und damit Tests durchführen können.</li> </ul>
Schulstufe	WPG (Wahlpflichtgegenstand, 6.-8. Klasse)
Nötiges Vorwissen	<p>Schleifen (for, while)</p> <p>Klassen</p> <p>Typumwandlungen (string, int, float)</p> <p>String Konkatenation</p> <p>Wertevergleich (&lt;, &gt;, =, !=), Minimum zweier Zahlen</p> <p>Methoden und Methodenaufrufe</p> <p>Benutzereingaben von Tastatur</p> <p>Ausnahmebehandlung (try, except)</p> <p>Einfache arithmetische Berechnungen (+, -, *, /, %)</p> <p>Modul time (sleep)</p> <p>Bonusaufgaben:</p> <p>Listen</p>
Bezug Kompetenzmodell	<p>Inhaltsdimension „Praktische Informatik“</p> <ul style="list-style-type: none"> <li>• Algorithmen, Datenstrukturen und Programmierung</li> </ul> <p>Handlungsdimension „Anwenden und Gestalten“</p> <ul style="list-style-type: none"> <li>• Aufgaben mit Mitteln der Informatik modellieren</li> <li>• Algorithmen entwerfen, implementieren und testen</li> <li>• Softwareprojekt planen und durchführen.</li> </ul> <p>Handlungsdimension „Reflektieren und Bewerten“</p>

	<ul style="list-style-type: none"> <li>• Gezielt nach Programmfehlern suchen und diese korrigieren</li> </ul>
Unterrichtsmethode	Einzelarbeit, Partnerarbeit, Plenum
Dauer	2 Doppelstunden
Anmerkungen	<p>Diese Aufgabe ist aufbauend auf Aufgabe 7.2. Die dort programmierte Methode <code>set_binary(number)</code> kommt zum Einsatz. Der Ablauf im Unterricht ist so gedacht, dass die Teilaufgabe 1 im Zweierteam bearbeitet wird. Anschließend folgt eine Diskussion der Ergebnisse im Rahmen der gesamten Gruppe. Diese Diskussion soll vor dem Bearbeiten der Teilaufgabe zwei eine einheitliche Ausgangslage schaffen. Die Lehrperson hat während der Diskussion die Möglichkeit, das Wissen der Schüler über Klassen, Erzeugen eines Objekts, ... „aufzufrischen“. Teilaufgabe zwei wird im Anschluss als Einzelarbeit gelöst.</p> <p>Die Teilaufgaben drei, vier und fünf sollen wieder im Zweierteam bearbeitet werden. Die Lehrperson kann bei Bedarf einzelnen Teams weiterhelfen oder mit der ganzen Klasse (fragend entwickelnd) versuchen, einen Lösungsweg zu erarbeiten. Die Ergebnisse der Teams sollen jedenfalls wieder diskutiert werden. Diese Vorgangsweise wird vorgeschlagen, um zu verhindern, dass Schüler Lösungswege einschlagen, die geringe Erfolgswahrscheinlichkeit haben. Die letzte Teilaufgabe (Programmieren) wird wieder als Einzelarbeit durchgeführt.</p> <p>Als Abschluss dieser Aufgabe sollte den Schülern Zeit gegeben werden, ihr Spiel zu benutzen – eventuell in Form eines Wettbewerbs in der Klasse.</p>

Tabelle 36: Binary Battle Lehrerinformation



### 7.3.2 Arbeitsblatt

#### Binary Battle

Du wirst das **Spiel** Binary\_Battle.py programmieren.

Hier sind die **Spielregeln**: Zu Beginn hat der Spieler null Punkte und drei Leben. Der Spieler bekommt vom Computer eine Zufallszahl für kurze Zeit am LED Strip Pidget angezeigt. Diese Zahl muss der Spieler in eine Dezimalzahl umrechnen und über die Tastatur eingeben.

Ist die eingegebene Zahl richtig, erhält der Spieler einen Punkt, ist sie falsch, verliert er ein Leben.

Nach fünf richtigen Antworten erhält der Spieler ein Zusatzleben und gelangt in den nächsten Level. Das Spiel beginnt mit maximal dreistelligen Binärzahlen (Level1), pro Level kommt eine Stelle dazu (bis maximal 8 Stellen).

Die Anzeigedauer der Zahl ist  $(1+Level/2)$  Sekunden – also für den Level 2 beispielsweise 2 Sekunden. Wenn alle Leben aufgebraucht sind, endet das Spiel.

So soll das Spiel auf der **Konsole** aussehen:

```
***** Welcome to Binary Battle *****

Please enter your name: Binary Checker

***** Start *****

What is your guess, Binary Checker? 3
Correct! You have 1 points!
What is your guess, Binary Checker? 6
Correct! You have 2 points!
[...]
What is your guess, Binary Checker? 4
Correct! You have 5 points!
New life for Binary Checker! 4 lives left.
New level is 2
[...]
What is your guess, Binary Checker? 12
False! Number was 10. You have 1 lives left.
What is your guess, Binary Checker? 14
False! Number was 15. You have 0 lives left.

Game over... you got 6 points.
```

**Aufgabe 1** Dein **Spieler** soll in der Klasse Player modelliert werden. Welche Felder außer dem Namen braucht ein Spieler und welche Methoden?  
 Felder: Methoden:

**Aufgabe 2** Programmiere die Klasse Player und teste die Methoden.

**Aufgabe 3** **Levels:** Das LED Strip Pidget hat acht LEDs mit denen es eine Binärzahl anzeigen kann. Dein Spiel soll mit der Zeit immer schwieriger werden, daher werden die gezeigten Zufallszahlen mit jedem Level größer. In der nächsten Tabelle sind für die ersten drei Levels die jeweils kleinste (oben) und die größte (unten) Zahl dargestellt. Rechne diese Zahlen in Dezimalzahlen um und vervollständige die Tabelle.

Level	Anzeige (min/max)	Kleinste Zahl	Größte Zahl
1			
2			
3			



### 7.3.3 Musterlösung

```
#!/usr/bin/env python

import LED_Strip as ledstrip
import time
import random

ledstrip.init()
number = 0
level = 1

class Player(object):
    def __init__(self,name):
        self.name = name
        self.points = 0
        self.lives = 3

    def add_point(self):
        self.points = self.points + 1

    def get_points(self):
        return self.points

    def add_life(self):
        self.lives = self.lives + 1

    def take_life(self):
        self.lives = self.lives - 1

    def get_lives(self):
        return self.lives

    def get_name(self):
        return self.name

def get_random(number, level):
    # level 1: numbers from 2 to 7
    # level 2: numbers from 4 to 15
    # level 3: numbers from 8 to 31 ...
    new_number = random.randint(2**level,2**(level+2)-1)

    # check that new number is not the same as one turn before
    while (number == new_number):
        new_number = random.randint(2**level,2**(level+2)-1)
    return new_number

def show_number(number, level):
    ledstrip.set_binary(number)
    time.sleep(1+level/2)
    ledstrip.set_binary(0)

def get_level(points):
    # min level=1, max level=6
    return min(6,int(points/5)+1)

def check_guess(number, guess):
    try:
        return number == int(guess)
    except ValueError:
        return False

print('***** Welcome to Binary Battle *****')

name = input('\nPlease enter your name: ')
player = Player(name)
```

```

print('\n***** Start *****\n')

while (player.get_lives()>0):
    number = get_random(number, level)
    show_number(number, level)
    guess = input('What is your guess, ' + player.get_name() + '? ')

    if check_guess(number, guess):
        player.add_point()
        print('Correct! You have ' + str(player.get_points()) + ' points!')
        # statement true, if player has n*5 points
        if get_level(player.get_points()) > get_level(player.get_points()-1):
            player.add_life()
            level = get_level(player.get_points())
            print('New life for '+player.get_name()+! '+str(player.get_lives())+' lives left.')
            print('New level is ' + str(level))
        else:
            player.take_life()
            print('False! Number was '+str(number)+'. You have '+str(player.get_lives())+' lives
left.')

print('\nGame over... you got ' + str(player.get_points()) + ' points.')

```

Listing 12: Binary\_Battle.py

### 7.3.4 Musterantworten

Aufgabe	Antwort
Aufgabe 3	Level 1: von 2 bis 7 Level 2: von 8 bis 15 Level 3: von 16 bis 31 $level_{max} = 6$ $number_{min} = 2^{level}$ $number_{max} = 2^{level+2}-1$

Tabelle 37: Binary Battle Musterantworten

## 7.4 Black Jack

### 7.4.1 Lehrerinformation

Verwendetes Pidget	Dice
Alltagsbezug	Würfelspiele sind den Schülern seit Kindestagen bekannt. Ähnlich ist es mit Kartenspielen. <i>Black Jack</i> (oder $17 + 4$ ) hat damit einen doppelten Alltagsbezug. Volljährige Schüler kennen das Spiel eventuell auch aus dem Spielcasino.
Motivation für die Aufgabe	Den Schülern soll bewusst werden, dass scheinbar einfache Abläufe (ich drücke auf eine Taste und irgendetwas passiert; ich drücke zweimal auf die Taste und etwas anderes passiert) im Detail nicht trivial zu realisieren sind.
Lernziele	<ul style="list-style-type: none"> <li>• Die Schüler sollen eine Aufgabe analysieren und in Teilaufgaben zerlegen können.</li> <li>• Die Schüler sollen Wissen über Hardware (Takt, Zustände, steigende/fallende Flanke) und Programmierkenntnisse vernetzen können.</li> <li>• Die Schüler sollen Wissen aus Mathematik (Wahrscheinlichkeitsrechnung) im Kontext der Aufgabenstellung anwenden.</li> <li>• Die Schüler sollen sinnvolle Testfälle konstruieren und damit Tests durchführen können.</li> </ul>
Schulstufe	WPG (Wahlpflichtgegenstand, 6.-8. Klasse)
Nötiges Vorwissen	<p>Schleifen (for, while)          Verzweigungen (if)          Klassen          Typumwandlungen (string, int, float)          String Konkatenation          Wertevergleich (&lt;, &gt;, ==, !=)          Methoden und Methodenaufrufe          Benutzereingaben von Tastatur          Einfache arithmetische Berechnungen (+, -, *, /, %)          Modul time (sleep, localtime)</p>
Bezug Kompetenzmodell	<p>Inhaltsdimension „Praktische Informatik“</p> <ul style="list-style-type: none"> <li>• Algorithmen, Datenstrukturen und Programmierung</li> </ul> <p>Handlungsdimension „Anwenden und Gestalten“</p> <ul style="list-style-type: none"> <li>• Aufgaben mit Mitteln der Informatik modellieren</li> <li>• Algorithmen entwerfen, implementieren und testen</li> <li>• Softwareprojekt planen und durchführen.</li> </ul> <p>Handlungsdimension „Reflektieren und Bewerten“</p> <ul style="list-style-type: none"> <li>• Gezielt nach Programmfehlern suchen und diese korrigieren</li> </ul>
Unterrichtsmethode	Einzelarbeit, Partnerarbeit, Plenum
Dauer	2 Doppelstunden
Anmerkungen	Der Ablauf im Unterricht ist so gedacht, dass die Teilaufgaben eins bis fünf im Zweierteam bearbeitet werden. Anschließend werden die Ergebnisse im Plenum diskutiert. Diese Diskussion soll vor dem Bearbeiten der Teilaufgabe sechs eine einheitliche Ausgangslage schaffen und sichern, dass die Schüler die Inhalte der ersten fünf

	<p>Teilaufgaben verstanden haben.</p> <p>Teilaufgabe sechs wird im Anschluss als Einzelarbeit gelöst. Die Lösungen und die Antworten auf die gestellten Fragen werden wieder im Plenum besprochen.</p> <p>Bei Teilaufgabe sieben werden zunächst in Partnerarbeit die benötigten Felder und Methoden erarbeitet, die Ergebnisse der Teams im Plenum besprochen und erst dann mit dem Programmieren begonnen. Diese Vorgangsweise wird vorgeschlagen, um zu verhindern, dass Schüler Lösungswege einschlagen, die geringe Erfolgswahrscheinlichkeit haben. Die letzte Teilaufgabe (Programmieren) wird wieder als Einzelarbeit durchgeführt.</p> <p>Als Abschluss dieser Aufgabe sollte den Schülern Zeit gegeben werden, ihr Spiel zu benutzen und verschiedene Spielerstrategien auszuprobieren.</p>
--	---

Tabelle 38: Black Jack Lehrerinformation

### 7.4.2 Arbeitsblatt

#### Black Jack

Du wirst das **Spiel** `Black_Jack.py` programmieren.

Hier sind die **Spielregeln**: Jeder Spieler muss versuchen durch Würfeln eine Summe zu erreichen, die so nahe wie möglich an 21 herankommt, 21 aber nicht überschreitet. Es gibt zwei Spieler. Spieler 1 ist der Benutzer des Programms, Spieler 2 ist der Computer. Für ein Spiel werden drei Runden gespielt, Gewinner ist, wer die meisten Runden gewonnen hat.

**Ablauf**: Spieler 1 wählt einen Spielernamen und startet den digitalen Würfel durch Drücken des Tasters am Dice Pidget. Die gewürfelte Zahl wird gespeichert. Spieler 1 kann diesen Vorgang durch nochmaliges Drücken des Tasters wiederholen, wobei die gewürfelten Zahlen zur ersten Zahl addiert werden. Ist der Spieler der Meinung, dass die erreichte Summe nahe genug an 21 ist, oder dass erneutes Würfeln zu einer Summe größer als 21 führen würde, dann beendet er durch rasches zweimaliges Drücken des Tasters (wie ein Doppelklick mit der Maus) seinen Spielzug.

Anschließend ist Spieler 2 (der Computer, Spielername *Raspi*) an der Reihe und muss versuchen, eine Summe zu erwürfeln, die näher an 21 liegt. Ein Unentschieden am Ende einer Runde ist nicht erlaubt.

So soll das Spiel auf der **Konsole** aussehen:

```
***** Welcome to Black Jack *****
```

```
Please enter your name: Blacky
```

```
***** Start *****
```

```
Player Blacky press switch to roll dice
```

```
Blacky has 6 points.
Blacky has 12 points.
Blacky has 17 points.
Blacky has 19 points.
```

```
Raspi has 6 points.
Raspi has 9 points.
Raspi has 14 points.
Raspi has 19 points.
Raspi has 22 points.
```

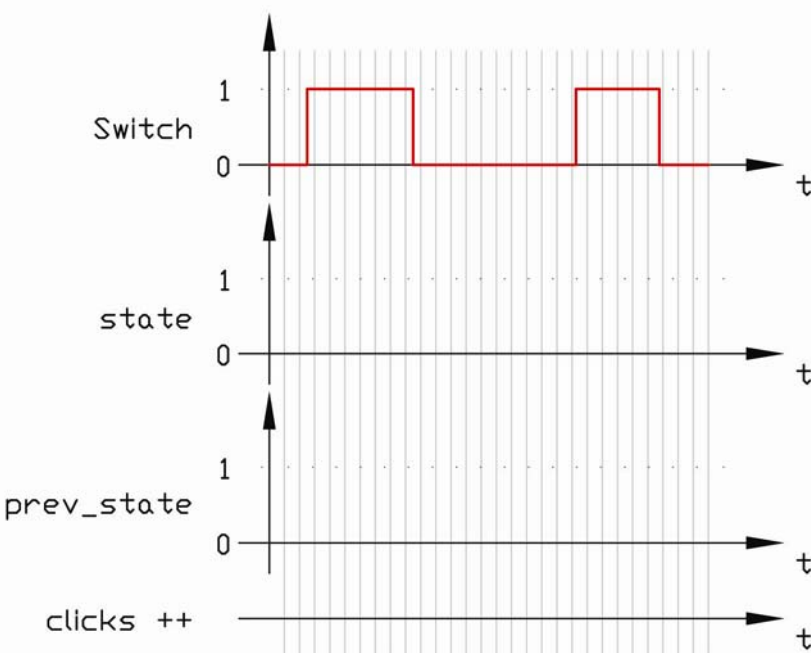
```
Blacky wins!
Score:
Blacky: 1
Raspi: 0
```

```
Player Blacky press switch to roll dice
[...]
```

```
Blacky wins!
Score:
Blacky: 2
Raspi: 1
```

Congratulations, you won!

Aufgabe 1	Notiere, welche Strategie Spieler 1 verfolgen muss. Ab welcher Summe ist es besser, nicht mehr zu würfeln?
Aufgabe 2	Notiere, welche Strategie der Computer verfolgen muss.

<p>Aufgabe 3</p>	<p>Notiere welche Spielsituationen es gibt, in denen bereits nach dem Spielzug von Spieler 1 der Ausgang der Runde feststeht?</p>
<p>Aufgabe 4</p>	<p>Die <b>Spieler</b> sollen in der Klasse <code>Player</code> modelliert werden. Welche Felder außer dem Namen braucht ein Spieler und welche Methoden?                  Felder: _____ Methoden: _____</p>
<p>Aufgabe 5</p>	<p><b>Statusabfrage des Tasters:</b>                  Dein Programm muss reagieren, wenn der Benutzer den Taster des Pidgets drückt. Es muss dabei unterscheiden, ob es sich um einen einfachen Klick (neue Zahl würfeln) oder einen Doppelklick (Spielzug beenden) handelt. Dazu muss das Programm während einer vorgegebenen Zeitdauer die Anzahl der Klicks zählen.                  Wie lange dauert ein Doppelklick? Wenn du auf deinem Mobiltelefon eine Stoppuhr hast, dann führe auf dem Start/Stop-Knopf der Stoppuhr einen Doppelklick aus. Wiederhole die Messung mehrfach. Welche Zeit hast du gestoppt?                  _____s</p> <p>Im folgenden Diagramm (Switch) siehst du den Zustand des Tasters am Pidget. Solange der Taster gedrückt ist, ist sein Zustand logisch 1, sonst logisch 0. Der Taster wurde also zweimal gedrückt. Die grauen vertikalen Linien stellen den Takt dar. Der Taster kann vom Benutzer zu jedem beliebigen Zeitpunkt gedrückt oder losgelassen werden. Die Variablen <code>state</code>, <code>prev_state</code> und <code>clicks</code> können ihren Wert jedoch nur genau zur Taktzeit ändern. Vervollständige die beiden Diagramme (<code>state</code> und <code>prev_state</code>), sodass die Variable <code>state</code> den Zustand des Tasters immer zur Taktzeit übernimmt. In <code>prev_state</code> wird der Zustand gespeichert, den <code>state</code> im vorangegangenen Taktzyklus hatte.</p>  <p>The diagram consists of four vertically stacked plots sharing a common horizontal time axis labeled 't'. The top plot, labeled 'Switch', shows a signal that is 0 most of the time but has two rectangular pulses that rise to 1. The second plot, labeled 'state', has a horizontal line at 0. The third plot, labeled 'prev_state', also has a horizontal line at 0. The bottom plot, labeled 'clicks ++', has a horizontal line at 0. Vertical grey lines represent clock ticks. The task is to complete the 'state' and 'prev_state' plots by updating their values at each tick to match the current state of the switch, and to mark the 'clicks' plot with vertical lines indicating when the switch transitions from 0 to 1.</p> <p>Die Variable <code>clicks</code> speichert, wie oft der Taster gedrückt wurde. Überlege, zu welchen Zeitpunkten die Variable um 1 erhöht werden muss und markiere diese Zeitpunkte auf der Zeitachse. Welchen Zustand haben <code>state</code> und <code>prev_state</code> zu diesen Zeitpunkten? <code>state</code> ist _____ und <code>prev_state</code> ist _____</p>
<p>Aufgabe 6</p>	<p>Als nächstes wirst du die Methode <code>get_clicks()</code> programmieren. Diese Methode liefert zurück, wie oft der Taster innerhalb einer vorgegebenen Dauer gedrückt wurde. Für das Diagramm von Aufgabe 5 müsste die Methode den Wert zwei zurückliefern. Auf der nächsten Seite siehst du ein Gerüst dieser Methode. Lies zuerst den Programmcode und die Kommentare und beantworte die Fragen Q1 bis Q3.</p>





### 7.4.3 Musterlösungen

```
#!/usr/bin/env python
import time
import random
import Dice as dice

dice.init()
clicks = 0
turns = 3

class Player(object):
    def __init__(self,name):
        self.name = name
        self.wins = 0
        self.points = 0

    def add_points(self, number):
        self.points = self.points + number

    def get_points(self):
        return self.points

    def set_points(self, number):
        self.points = number

    def add_wins(self):
        self.wins = self.wins + 1

    def get_wins(self):
        return self.wins

    def get_name(self):
        return self.name

def get_clicks():
    # waiting for first click
    while not dice.get_switch():
        time.sleep(0.05)
    state = True
    prev_state = True
    clicks = 1
    first_click = time.time()
    # 0.3 seconds to do another click
    while time.time() < first_click + 0.3:
        prev_state = state
        state = dice.get_switch()
        # GPIO Input rising slope
        if state and not prev_state:
            clicks = clicks + 1
        time.sleep(0.015)
    return clicks

def roll_dice():
    # to generate a more realistic impression of a rolling dice, the pidget
    # shows i different numbers
    i = random.randint(10,20)
    for k in range(i):
        # rolling: dice makes a circular count from 1 to 6 and stops after i steps
        number = k%6 + 1
        dice.set_dice(number)
        # rolling becomes slower as k increases
        time.sleep(0.015*k)
    return number

print('***** Welcome to Black Jack *****')
```

```

raspi = Player('Raspi')
name = input('\nPlease enter your name: ')
player = Player(name)

print('\n***** Start *****')

for i in range (0,turns):
    print('\nPlayer ' + player.get_name() + ' press switch to roll dice\n')
    clicks = get_clicks()
    while clicks < 2 and player.get_points() <= 21:
        if clicks == 1:
            number = roll_dice()
            player.add_points(number)
            print(player.get_name() + ' has ' + str(player.get_points()) + ' points.')
            clicks = get_clicks()

    if player.get_points() < 21:
        print()
        dice.set_dice(0)
        time.sleep(1)
        while raspi.get_points() <= player.get_points():
            number = roll_dice()
            raspi.add_points(number)
            print(raspi.get_name() + ' has ' + str(raspi.get_points()) + ' points.')
            time.sleep(2)

    if player.get_points() == 21 or raspi.get_points() > 21:
        print('\n' + player.get_name() + ' wins!')
        player.add_wins()
    elif player.get_points() > 21 or raspi.get_points() > player.get_points():
        print('\n' + raspi.get_name() + ' wins!')
        raspi.add_wins()
    print('Score:')
    print(player.get_name() + ': ' + str(player.get_wins()))
    print(raspi.get_name() + ': ' + str(raspi.get_wins()) + '\n')

    clicks = 0
    dice.set_dice(0)
    player.set_points(0)
    raspi.set_points(0)
    if player.get_wins()>raspi.get_wins():
        print('Congratulations, you won!')
    else:
        print('I am sorry, you lost!')

```

Listing 13: Black\_Jack.py

```

#!/usr/bin/env python
import random

score = 0
wins = 0
tests = 10000

for i in range (0,tests):
    while score <= 18:
        score = score + random.randint(1,6)
        if score < 22:
            wins = wins +1
        score = 0

print('raspi wins: ' + str(wins))
print('player wins: ' + str(tests-wins))

```

Listing 14: Black\_Jack\_Test.py

7.4.4 Musterantworten

Aufgabe	Antwort														
Aufgabe 1	<p>Bei einem Punktestand von 19 oder 20 Punkten sollte der Spieler aufhören, weil die Wahrscheinlichkeit nicht mehr als 21 Punkte zu erreichen bei <math>1/3</math> bzw. <math>1/6</math> liegt. Bei weniger als 18 Punkten sollte noch einmal gewürfelt werden, weil die Wahrscheinlichkeit einen Punktestand kleiner oder gleich 21 zu erreichen größer als 50% ist. Bei einem Punktestand von 18 Punkten liegt diese Wahrscheinlichkeit genau bei 50%. Allerdings muss in diesem Fall die Gewinnwahrscheinlichkeit für den Gegenspieler (Computer) betrachtet werden.</p> <p>In Tabelle 39 wurden folgende Bezeichnungen gewählt: <math>i</math> ist der Ausgangspunktestand des Computers. <math>G_i</math> ist das Ereignis, dass der Computer ausgehend vom Punktestand <math>i</math> das Spiel gewinnt. <math>P(G_i)</math> ist die Wahrscheinlichkeit, dass <math>G_i</math> eintritt. <math>X</math> bezeichnet die gewürfelte Augenzahl.</p> <p>Die Berechnungen wurden unter der Annahme durchgeführt, dass der Spieler bei 18 Punkten steht und die Punktestände 13 bis 18 mit gleicher Wahrscheinlichkeit erreicht werden.</p> <p>Es ist aus Tabelle 39 ersichtlich, dass die Gewinnwahrscheinlichkeit <math>P(G_i)</math> in den aufgelisteten Fällen bei mindestens 50% liegt. Das bedeutet umgekehrt, dass der Spieler bei einem Punktestand von 18 Punkten noch einmal würfeln soll.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Fall</th> <th>Gewinnwahrscheinlichkeit des Computers</th> </tr> </thead> <tbody> <tr> <td>18 Punkte</td> <td><math>P(G_{18}) = P(X = 1) + P(X = 2) + P(X = 3) = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = 0,5</math></td> </tr> <tr> <td>17 Punkte</td> <td><math>P(G_{17}) = P(X = 1) \cdot P(G_{18}) + P(2 \leq X \leq 4) = \frac{1}{6} \cdot 0,5 + 0,5 = 0,58</math></td> </tr> <tr> <td>16 Punkte</td> <td><math>P(G_{16}) = 0,68</math></td> </tr> <tr> <td>15 Punkte</td> <td><math>P(G_{15}) = 0,79</math></td> </tr> <tr> <td>14 Punkte</td> <td><math>P(G_{14}) = 0,76</math></td> </tr> <tr> <td>13 Punkte</td> <td><math>P(G_{13}) = 0,72</math></td> </tr> </tbody> </table> <p style="text-align: center;">Tabelle 39: Gewinnwahrscheinlichkeit Computer</p> <p>Die Simulation von 10000 Spielverläufen, bei denen der Spieler bei 18 Punkten steht und der Computer am Zug ist, brachte ein Verhältnis von ca. 71:29 für den Computer.</p>	Fall	Gewinnwahrscheinlichkeit des Computers	18 Punkte	$P(G_{18}) = P(X = 1) + P(X = 2) + P(X = 3) = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = 0,5$	17 Punkte	$P(G_{17}) = P(X = 1) \cdot P(G_{18}) + P(2 \leq X \leq 4) = \frac{1}{6} \cdot 0,5 + 0,5 = 0,58$	16 Punkte	$P(G_{16}) = 0,68$	15 Punkte	$P(G_{15}) = 0,79$	14 Punkte	$P(G_{14}) = 0,76$	13 Punkte	$P(G_{13}) = 0,72$
Fall	Gewinnwahrscheinlichkeit des Computers														
18 Punkte	$P(G_{18}) = P(X = 1) + P(X = 2) + P(X = 3) = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = 0,5$														
17 Punkte	$P(G_{17}) = P(X = 1) \cdot P(G_{18}) + P(2 \leq X \leq 4) = \frac{1}{6} \cdot 0,5 + 0,5 = 0,58$														
16 Punkte	$P(G_{16}) = 0,68$														
15 Punkte	$P(G_{15}) = 0,79$														
14 Punkte	$P(G_{14}) = 0,76$														
13 Punkte	$P(G_{13}) = 0,72$														
Aufgabe 2	Strategie des Computers ist, so lange zu würfeln wie sein Punktestand kleiner oder gleich dem Punktestand des Spielers ist.														
Aufgabe 3	Punktestand = 21 → Spieler hat gewonnen Punktestand > 21 → Computer hat gewonnen														
Aufgabe 5	Dauer eines Doppelklicks ca. 0,2s; in der Musterlösung wurde ein etwas größerer Zeitraum gewählt (0,3s), damit der Doppelklick zuverlässig funktioniert. Es hat sich bei dieser Aufgabe gezeigt, dass bei Drücken bzw. Loslassen des Tasters wirklich jeweils nur eine Zustandsänderung passiert und das in Kapitel 5.4 beschriebene Entprellen des Tasters funktioniert. <b>Fehler! Verweisquelle konnte nicht gefunden werden.</b> zeigt das vollständige Timingdiagramm für zwei Klicks.														

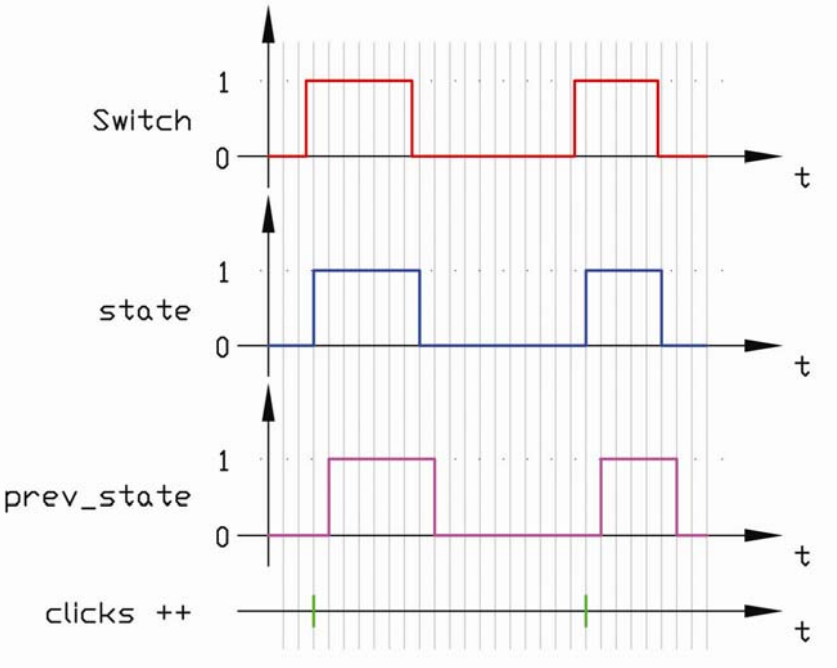
	 <p style="text-align: center;">Abbildung 41: Vollständiges Timingdiagramm</p> <p>Zustände: <code>state=1</code>, <code>prev_state=0</code>; Dieser Zustand ist immer dann erreicht, wenn der Schalter und in der Folge <code>state</code> von 0 auf 1 übergeht (steigende Flanke).</p>
<p>Aufgabe 6</p>	<p>Q1: <code>time.sleep(0.05)</code> verhindert <i>busy waiting</i>. Der Taster wird mit einer Frequenz <math>f = \frac{1}{T} = \frac{1}{0,05} = 20\text{Hz}</math> abgefragt.</p> <p>Q2: <code>clicks</code> wird mit 1 initialisiert, weil an dieser Stelle der Taster bereits einmal gedrückt wurde.</p> <p>Q3: vgl. Q1: Frequenz <math>f = \frac{1}{T} = \frac{1}{0,015} \sim 67\text{Hz}</math></p>

Tabelle 40: Black Jack Musterantworten

## 7.5 Clock

### 7.5.1 Lehrerinformation

Verwendetes Pidget	7 Segment Display
Alltagsbezug	Schuluhr, Uhr am Mobiltelefon, Wecker, Wanduhr, Uhren auf Backöfen, Dampfgeräten, Satellitenempfängern, Autoradios, Stereoanlagen, ... Uhren sind überall, wo auch die Schüler sind.
Motivation für die Aufgabe	Motivation war, mit einer überschaubaren Aufgabe die Behandlung von Tasteneingaben am Pidget zu erarbeiten und als Endergebnis ein funktionierendes Gerät zu haben. Auch die Modulrechnung als Grundrechnungsart in der Informatik soll mit dieser Aufgabe geübt werden.
Lernziele	<ul style="list-style-type: none"> <li>• Die Schüler sollen eine Aufgabe analysieren und in Teilaufgaben zerlegen können.</li> <li>• Die Schüler sollen Wissen über Hardware (Takt, Zustände, steigende/fallende Flanke) und Programmierkenntnisse vernetzen können.</li> <li>• Die Schüler sollen sinnvolle Testfälle konstruieren und damit Tests durchführen können.</li> <li>• Die Schüler sollen den Einfluss einzelner Parameter im Programmablauf bewerten können.</li> </ul>
Schulstufe	WPG (Wahlpflichtgegenstand, 6.-8. Klasse)
Nötiges Vorwissen	Schleifen (for, while) Verzweigungen (if) Wertevergleich (<, >, =, !=) Einfache arithmetische Berechnungen (+, -, *, /, %) Modul time (sleep, localtime)
Bezug Kompetenzmodell	Inhaltsdimension „Praktische Informatik“ <ul style="list-style-type: none"> <li>• Algorithmen, Datenstrukturen und Programmierung</li> </ul> Handlungsdimension „Anwenden und Gestalten“ <ul style="list-style-type: none"> <li>• Aufgaben mit Mitteln der Informatik modellieren</li> <li>• Algorithmen entwerfen, implementieren und testen</li> </ul> Handlungsdimension „Reflektieren und Bewerten“ <ul style="list-style-type: none"> <li>• Gezielt nach Programmfehlern suchen und diese korrigieren</li> </ul>
Unterrichtsmethode	Einzelarbeit, Plenum
Dauer	2 Doppelstunden
Anmerkungen	Die Teilaufgaben des Arbeitsblattes sind in Einzelarbeit zu erledigen. Etappenziele können bei Bedarf im Plenum diskutiert und erklärt werden, damit leistungsschwächere Schüler ebenfalls die Ziele der Aufgabe erreichen.

Tabelle 41: Clock Lehrerinformation

### 7.5.2 Arbeitsblatt

<b>Clock</b>															
<p>Deine Aufgabe ist, das 7 Segment Display zu einer Uhr (Clock.py) zu machen. Die Uhr hat 3 Betriebsmodi. In Modus 0 wird die Uhrzeit angezeigt, in Modus 1 die Sekunden und in Modus 2 das Datum. Durch drücken des Tasters am 7 Segment Pidget wechselt die Uhr in den nächsten Modus. Nach Modus 2 folgt wieder Modus 0. Die folgende Tabelle zeigt, was die vier Stellen des 7 Segment Display – abhängig vom Betriebsmodus – anzeigen sollen:</p>															
<table border="1" style="margin: auto; border-collapse: collapse;"> <thead> <tr> <th rowspan="2" style="padding: 5px;">Modus</th> <th colspan="2" style="padding: 5px;">7 Segment Display</th> </tr> <tr> <th style="padding: 5px;">Linke Hälfte</th> <th style="padding: 5px;">Rechte Hälfte</th> </tr> </thead> <tbody> <tr> <td style="text-align: center; padding: 5px;">0</td> <td style="padding: 5px;">Stunden</td> <td style="padding: 5px;">Minuten</td> </tr> <tr> <td style="text-align: center; padding: 5px;">1</td> <td style="padding: 5px;">ausgeschaltet</td> <td style="padding: 5px;">Sekunden</td> </tr> <tr> <td style="text-align: center; padding: 5px;">2</td> <td style="padding: 5px;">Tag</td> <td style="padding: 5px;">Monat</td> </tr> </tbody> </table>		Modus	7 Segment Display		Linke Hälfte	Rechte Hälfte	0	Stunden	Minuten	1	ausgeschaltet	Sekunden	2	Tag	Monat
Modus	7 Segment Display														
	Linke Hälfte	Rechte Hälfte													
0	Stunden	Minuten													
1	ausgeschaltet	Sekunden													
2	Tag	Monat													
<p>Das Programm soll endlos laufen (Endlosschleife), also bis es vom Benutzer durch Strg + C unterbrochen wird.</p>															
Aufgabe 1	<p><b>Abfrage des Tasters</b> Dein Programm läuft in einer Endlosschleife. Es muss bei jedem Schleifendurchlauf überprüfen, ob der Taster am Pidget gedrückt ist. Falls er gedrückt ist, muss die Uhr den Modus wechseln. Ist der Taster beim nächsten Schleifendurchlauf immer noch gedrückt, darf die Uhr allerdings den Modus nicht noch einmal wechseln, sondern erst wieder, nachdem der Taster losgelassen und erneut gedrückt wurde. Das Programm muss also nicht nur den Zustand des Schalters (<code>state</code>) berücksichtigen, sondern auch, welchen Zustand der Schalter im vorangegangenen Schleifendurchlauf hatte (<code>prev_state</code>). Überlege und notiere: Von welchem Datentyp sollen die Variablen <code>state</code> und <code>prev_state</code> sein (Begründe!)? Welchen Wert müssen <code>state</code> und <code>prev_state</code> haben, damit ein Moduswechsel durchgeführt werden darf?</p> <p>Datentyp: _____                      Weil: _____</p> <p><code>state = _____</code>                      <code>prev_state = _____</code></p> <p>Schreibe ein Programm <code>Switch_Test.py</code>, das in einer Endlosschleife läuft und die Variablen <code>state</code> und <code>prev_state</code> verwendet. Das Programm soll bei jedem Betätigen des Tasters auf der Konsole genau einmal den Text ‚Taster gedrückt‘ ausgeben.  <b>Implementierungstipps:</b> Das 7 Segment Pidget hat eine Methode, die den Zustand des Tasters zurückliefert. Lies dazu die <code>description()</code> des 7 Segment Moduls.</p>														
Aufgabe 2	<p><b>Weiterschalten des Modus:</b> Die Stelle, an der in <code>Switch_Test.py</code> die <code>print()-</code>Anweisung steht, ist genau die Stelle, an der der Modus weitergeschaltet werden muss. Es müssen hier der Variable <code>mode</code> nacheinander die Werte 0, 1, 2, 0, 1, 2, ... zugewiesen werden. Von welchem Datentyp muss <code>mode</code> sein? Typ: _____ Ergänze dein Programm so, dass diese Zuweisungen erfolgen. Lass dir den Wert der Variablen <code>mode</code> auf der Konsole ausgeben und teste ob dein Programm funktioniert.</p>														
Aufgabe 3	<p><b>Modus 1:</b> Implementiere den ersten Modus, damit das 7 Segment Pidget die Uhrzeit anzeigt.  <b>Implementierungstipps:</b> <code>time.localtime().tm_hour</code> liefert die aktuelle Stunde. Andere Elemente liefern <code>tm_mon</code>, <code>tm_mday</code>, <code>tm_min</code>, <code>tm_sec</code></p>														
Aufgabe 4	<p><b>Modi 2 und 3:</b> Implementiere die restlichen Modi. Im Modus 2 (Sekunden anzeigen) musst du die beiden linken Anzeigen ausschalten. Das erreichst du indem du jeder dieser Stellen eine Zahl größer als 9 anzeigen lässt.</p>														
Aufgabe 5	<p>Starte dein Programm und beobachte in der rechten unteren Bildschirmecke die Auslastung der CPU – sie steigt auf 100%.</p>														

	<p>Warum ist das so?</p> <p>Um sie zu entlasten musst du an geeigneter Stelle die Anweisung <code>time.sleep(x)</code> einfügen. Experimentiere mit verschiedenen Werten von <code>x</code> und beobachte wie sich die CPU Auslastung und die Reaktionszeit auf einen Tastendruck am Pidget verändern. Notiere deine Beobachtungen:</p> <p>Welchen Wert für <code>x</code> hast du gewählt? <code>x</code> = _____s</p>
<p>Bonus 1</p>	<p>Der <b>wichtigste Modus</b> deiner Uhr ist die Anzeige der Uhrzeit. Verändere das Programm so, dass der Modus 2 oder der Modus 3 nach fünf Sekunden automatisch beendet wird und das Programm in Modus 1 wechselt.</p>
<p>Bonus 2</p>	<p><b>Showtime:</b> Verändere dein Programm so, dass die Uhr zu jeder vollen Stunde etwas Ungewöhnliches anzeigt. Es kann zum Beispiel der Displaytest des 7 Segment Displays ausgeführt werden.</p>



### 7.5.3 Musterlösungen

```
#!/usr/bin/env python
import time
import Seven_Segment_Display as display

display.init()

state = False
prev_state = False
mode = 0
click_time = 0

while True:
    state = display.get_switch()
    # GPIO Input rising slope
    if state and not prev_state:
        mode = (mode+1)%3
        click_time = time.time()
        prev_state = state

    # there is no switch-case statement in Python
    if mode == 0:
        display.set_double_digit(0,time.localtime().tm_min)
        display.set_double_digit(1,time.localtime().tm_hour)
    elif mode == 1:
        display.set_double_digit(0,time.localtime().tm_sec)
        # sets digits 2 and 3 to blank
        display.set_digit(2,10)
        display.set_digit(3,10)
    else:
        display.set_double_digit(0,time.localtime().tm_mon)
        display.set_double_digit(1,time.localtime().tm_mday)

    # display time after 5 seconds
    if mode != 0 and time.time() > click_time + 5:
        mode = 0

    # perform display test every hour
    if time.localtime().tm_sec == 0 and time.localtime().tm_min == 0:
        display.test()

    # reduce cpu load
    time.sleep(0.05)
```

Listing 15: Clock.py

```
#!/usr/bin/env python
import time
import Seven_Segment_Display as display

display.init()

state = False
prev_state = False

while True:
    state = display.get_switch()
    # GPIO Input rising slope
    if state and not prev_state:
        print('switch pressed')
        prev_state = state
```

Listing 16: Switch\_Test.py

### 7.5.4 Musterantworten

Aufgabe	Antwort
Aufgabe 1	Die Variablen <code>state</code> und <code>prev_state</code> sind vom Typ Boolean. Eine steigende Flanke am Taster ist dann gegeben, wenn <code>state=1</code> und <code>prev_state=0</code> ist.
Aufgabe 2	Datentyp für <code>mode</code> ist Integer.
Aufgabe 5	Die CPU geht auf Volllast, weil die Schleife ununterbrochen wiederholt wird (Endlosschleife). <code>time.sleep(x)</code> ist die letzte Anweisung im Schleifenblock. Bei $x > 0.2s$ gibt es spürbare Verzögerungen zwischen dem Tastendruck und dem Wechseln des Modus, speziell, wenn die Taste zweimal gedrückt wird. Für die Musterlösung wurde der Wert $x = 0.05s$ gewählt. Bei diesem Wert werden auch rasche Doppelclicks fehlerfrei umgesetzt und die CPU-Auslastung ist wie im Leerlauf.

Tabelle 42: Clock Musterantworten

## 7.6 Morse Transmitter

### 7.6.1 Lehrerinformation

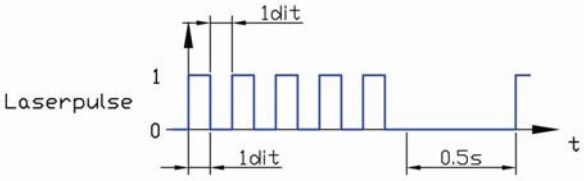
Verwendetes Pidget	Optical Transmitter
Alltagsbezug	Die Wikipedia-Seite über den Morsecode [33] listet unter „Sonstiges“ einige Anwendungen für Morsecode auf, über die sich ein Alltagsbezug für die Schüler herstellen lässt. Exemplarisch sei hier der Jingle der ZDF Nachrichtensendung „heute“ genannt, in dem der Morsecode für das Wort „heute“ den Rhythmus des Jingles vorgibt. [34]
Motivation für die Aufgabe	Nachrichten beim Sender zu kodieren und beim Empfänger zu dekodieren ist immer spannend. Hier ist wird die kodierte Nachricht nicht über elektrische Leitungen übertragen, sondern sichtbar über optische Signale. Morsecode wurde aus mehreren Gründen gewählt: Zunächst ist die Codierung einfach zu verstehen. Es gibt nur zwei Symbole (Punkt und Strich) und drei Regeln für Länge der Pausen (zwischen zwei Symbolen / Buchstaben / Wörtern). Durch die unterschiedlich langen Pausen kann sich der Empfänger bei Übertragungsfehlern neu synchronisieren. Der dritte Grund für die Wahl des Morsecodes ist, dass bei entsprechend langsamer Übertragung die Schüler mitlesen können, was ihr Programm überträgt. Auch der Zusammenhang zwischen der Häufigkeit, mit der ein Zeichen in der Sprache vorkommt und der Codelänge für dieses Zeichen kann beim Morsecode schön gezeigt werden.
Lernziele	<ul style="list-style-type: none"> <li>• Die Schüler sollen eine komplexe Aufgabe analysieren, in Teilaufgaben zerlegen und arbeitsteilig lösen können.</li> <li>• Die Schüler sollen für die Symbole einer Codierung (hier: Punkt und Strich) und die daraus gebildeten Zeichen eine für die Anwendung geeignete Struktur entwerfen und implementieren können.</li> <li>• Die Schüler sollen sinnvolle Testfälle konstruieren und damit an Aufgabenteilen Tests durchführen können.</li> </ul>
Schulstufe	WPG (Wahlpflichtgegenstand, 6.-8. Klasse)
Nötiges Vorwissen	Morsecodierung Listen (Dictionary) Schleifen (for, while) Verzweigungen (if) Wertevergleich (<, >, ==, !=) Einfache arithmetische Berechnungen (+, -, *, /, %) Modul time (sleep, time)
Bezug Kompetenzmodell	Inhaltsdimension „Praktische Informatik“ <ul style="list-style-type: none"> <li>• Algorithmen, Datenstrukturen und Programmierung</li> </ul> Handlungsdimension „Anwenden und Gestalten“ <ul style="list-style-type: none"> <li>• Aufgaben mit Mitteln der Informatik modellieren</li> <li>• Algorithmen entwerfen, implementieren und testen</li> </ul> Handlungsdimension „Reflektieren und Bewerten“ <ul style="list-style-type: none"> <li>• Gezielt nach Programmfehlern suchen und diese korrigieren</li> </ul>

Unterrichtsmethode	Teamarbeit, Plenum
Dauer	2 Doppelstunden
Anmerkungen	<p>Bei dieser und der dazugehörigen Aufgabe 7.7 ist es besonders wichtig, den Schülern zu Beginn das fertige System vorzuführen. Es wird vorgeschlagen, zuerst wenig Text bei langsamer Übertragungsgeschwindigkeit zu übermitteln und dann eine Übertragung mit Maximalgeschwindigkeit zu demonstrieren. Es bietet sich an, diese Aufgabe (auch wieder gemeinsam mit Aufgabe 7.7) als Wettbewerb zwischen den Teams zu geben. Ziel ist, am Ende einen vorgegebenen Text möglichst rasch und fehlerfrei zu übertragen.</p> <p>Damit die Schüler den Stand der Entwicklung testen können, kann die Musterlösung des Morse Code Receivers in den Testmodus geschaltet werden. Dazu muss in Zeile 8 die Variable <code>test_mode</code> mit <code>True</code> anstatt mit <code>False</code> initialisiert werden. Der Receiver läuft dann mit der fixen Empfangsgeschwindigkeit von 50 Dits/Sekunde und beendet bei längeren Pausen (länger als 15 Dits) nicht den Empfang. Der Ablauf dieser Aufgabe soll so sein, dass sich zu Beginn die Zweierteams überlegen, wie die Übersetzung von Zeichen in Morsecode erfolgen soll. Die Ergebnisse der Diskussion sollen im Plenum diskutiert und abgeglichen werden. Wenn der Datentyp Dictionary (so wie in der Musterlösung) zum Einsatz kommt, ist es für die Schüler hilfreich, wenn sie die Übersetzungstabelle aus der Musterlösung zur Verfügung gestellt bekommen.</p> <p>Die restlichen Teilaufgaben sollen im Zweierteam bearbeitet werden.</p> <p>Als Zeichensatz ist die Schnittmenge aus Morsecode und ASCII Code vorgegeben, um zusätzliche Schwierigkeiten bei der Implementierung zu vermeiden. Eine Erweiterung des Zeichensatzes auf Umlaute könnte als Zusatzaufgabe gestellt werden.</p>

Tabelle 43: Morse Transmitter Lehrerinformation

### 7.6.2 Arbeitsblatt

<b>Morse Transmitter</b>	
<p>Deine Aufgabe ist es, den Optical Transmitter so zu programmieren (<code>Morse_Transmitter.py</code>), dass er einen vom Benutzer eingegebenen Text als morsecodierte Laserimpulse absendet. Für den Eingabetext sind folgenden Zeichen gültig:</p> <p style="text-align: center;">A ... Z, a ... z, 0 ... 9, ( ) + , - . / : ; = ? @</p> <p>Kleinbuchstaben werden wie der dazugehörige Großbuchstabe übertragen, ungültige Eingabezeichen sollen gemeinsam mit einer Fehlermeldung am Bildschirm ausgegeben werden.</p>	
Aufgabe 1	<p>Das Programm muss den Eingabetext in Morsecode übersetzen. Diskutiere mit deinem Partner in welchem Datentyp du die Übersetzungstabelle zwischen Zeichen und Morsecode speichern würdest. Notiere das Ergebnis eurer Diskussion hier:</p> <p>Der Morsecode besteht aus zwei Symbolen, dem Punkt und dem Strich, die sich bei der Übertragung durch unterschiedliche Impulslängen unterscheiden. Wie würdet ihr im oben gewählten Datentyp den Punkt und den Strich repräsentieren?</p> <p>Punkt als _____ Strich als _____</p> <p>Wie würde eure Übersetzungstabelle aussehen, wenn sie nur aus den Buchstaben C (Morsecode <code>-.-</code>) und D (<code>-..</code>) bestünde?</p> <p>code =</p>
Aufgabe 2	<p>Implementiere die Methode <code>transmit_char(c)</code>, der ein beliebiges Zeichen <code>c</code> übergeben wird und die dieses über Laserpulse im Morsecode überträgt. Lies die <code>description()</code> des Optical Transmitter Moduls und wähle eine geeignete Methode für das Übertragen der Impulse aus.</p> <p>Die Methode muss folgende Dinge berücksichtigen:</p> <ul style="list-style-type: none"> <li>• Ist das übergebene Zeichen in der Übersetzungstabelle?             <ul style="list-style-type: none"> <li>○ Falls ja: Zeichen übertragen</li> <li>○ Falls Leerzeichen: in der Übertragung entsprechende Pause einhalten</li> <li>○ Sonst: Zeichen und Fehlermeldung am Bildschirm ausgeben</li> </ul> </li> <li>• „Dahs“ sind dreimal so lang wie „Dits“ – die Laserpulse müssen es auch sein.</li> <li>• Zwischen zwei Zeichen muss eine Pause mit der Länge von 3 „Dits“ sein.</li> <li>• Zwischen zwei Wörtern muss eine Pause mit der Länge von 7 „Dits“ sein.</li> </ul> <p>Überlege, wie ein Teststring aussehen muss, der die obigen Fälle abdeckt und schreibe dir diesen hardcodiert in dein Programm. Übergib den String Zeichen für Zeichen an die Methode. Wähle die Dauer eines Dits so, dass du kontrollieren kannst, ob die Abfolge der Laserpulse korrekt ist. Wenn du der Meinung bist, dass die Methode fehlerfrei funktioniert, dann wähle die Dauer für ein Dit gleich 20ms und teste deine Übertragung am bereitgestellten Optical Receiver Pidget.</p>
Aufgabe 3	<p>Erweitere dein Programm so, dass der Benutzer den zu übertragenden Text über die Tastatur eingeben kann. Dabei soll jedes Zeichen, das übertragen wurde, auch am Bildschirm ausgegeben werden.</p> <p>Für die zeichenweise Ausgabe am Bildschirm kannst du folgende Befehle verwenden:</p> <pre>import sys  sys.stdout.write(c) sys.stdout.flush()</pre>

	<p>Die Ausgabe auf der Konsole soll so aussehen:</p> <pre> ***** Morse Code Transmitter *****  Enter text to be transmitted: Das ist ein Probetext.  ***** Start of transmission *****  DAS IST EIN PROBETEXT.  ***** End of transmission ***** </pre> <p>Beachte, dass für die Übertragung und für die Bildschirmausgabe Kleinbuchstaben in Großbuchstaben umgewandelt wurden.</p>
<p>Aufgabe 4</p>	<p>Damit bei längeren Texten sinnvolle Zeilenumbrüche entstehen, erweitere deine Textausgabe so, dass beim ersten Leerzeichen ab dem 65. Zeichen einer Zeile, ein Zeilenumbruch gemacht wird.</p>
<p>Aufgabe 5</p>	<p>Veränderbare Übertragungsgeschwindigkeit: Erweitere dein Programm so, dass der Benutzer die Übertragungsgeschwindigkeit in Dits/s in den Grenzen [1,100] eingeben kann. Bei Eingabe ungültiger Werte soll der Benutzer zu einer erneuten Eingabe aufgefordert werden. Die Bildschirmausgabe soll so aussehen:</p> <pre> ***** Morse Code Transmitter *****  Enter speed of transmission [1,100] dits/s: 150 Speed not valid, please reenter [1,100] dits/s: 75  Enter text to be transmitted: Das ist ein Probetext </pre>
<p>Aufgabe 6</p>	<p>Zusammenfassung: Erweitere dein Programm so, dass am Ende der Übertragung die Anzahl der übertragenen Zeichen und die Übertragungsdauer angezeigt werden. Die Ausgabe soll so aussehen:</p> <pre> ***** End of transmission *****  22 characters transmitted in 3.64 s.  ***** </pre>
<p>Bonus 1</p>	<p>Synchronisieren der Übertragungsgeschwindigkeit: Dein Programm soll vor dem Übertragen der Daten die folgende Sequenz an den Empfänger schicken, damit sich dieser auf die Übertragungsgeschwindigkeit synchronisieren kann:</p> <div style="text-align: center;">  </div> <p>Die Sequenz besteht aus 5 Laserpulsen und Pausen, die die Länge von je einem Dit haben. Dann folgt eine Pause von 0,5s und dann beginnt die Datenübertragung. Die Bildschirmausgabe soll so aussehen:</p> <pre> ***** Morse Code Transmitter *****  Enter speed of transmission [1,100] dits/s: 50  Enter text to be transmitted: Das ist ein Probetext.  Setting up speed of transmission to 50 dits/s.  ***** Start of transmission ***** </pre>

### 7.6.3 Musterlösungen

```
#!/usr/bin/env python
import time
import sys
import Optical_Transmitter as transmitter

transmitter.init()

# Morse code:
# symbols:
# '.' dit, length = 1 * dit
# '-' Dah, length = 3 * dit
#
# break between symbols: 1 dit
# break between characters: 3 dits
# break between words: 7 dits

# Dictionnary of Morse codes for: ( ) + , - . / 0 ... 9 : ; = ? @ A ... Z
code = {'(':[3,1,3,3,1],')':[3,1,3,3,1,3,3],'+':[1,3,1,3,1],',':[3,3,1,1,3,3],'-':
:[3,1,1,1,3,3],'.':[1,3,1,3,1,3,3],'/':[3,1,1,3,1], '0':[3,3,3,3,3], '1':[1,3,3,3,3,3], '2':[1,1,
3,3,3], '3':[1,1,1,3,3,3], '4':[1,1,1,1,3,3], '5':[1,1,1,1,1,1], '6':[3,1,1,1,1], '7':[3,3,1,1,1], '8':
[3,3,3,1,1], '9':[3,3,3,3,1], ':':[3,3,3,1,1,1], ';':[3,1,3,1,3,1], '=':[3,1,1,1,3,3], '?':[1,1,3,
3,1,1], '@':[1,3,3,1,3,1], 'A':[1,3], 'B':[3,1,1,1], 'C':[3,1,3,1], 'D':[3,1,1], 'E':[1], 'F':[1,1,
3,1], 'G':[3,3,1], 'H':[1,1,1,1], 'I':[1,1], 'J':[1,3,3,3], 'K':[3,1,3], 'L':[1,3,1,1], 'M':[3,3]
, 'N':[3,1], 'O':[3,3,3], 'P':[1,3,3,1], 'Q':[3,3,1,3], 'R':[1,3,1], 'S':[1,1,1], 'T':[3], 'U':[1,1,
3], 'V':[1,1,1,3], 'W':[1,3,3], 'X':[3,1,1,3], 'Y':[3,1,3,3], 'Z':[3,3,1,1]}

def transmit_char(c):
    if c in code:
        # transmit character
        for i in code[c]:
            transmitter.set_laserpulse(int(i)*dit)
            # break between symbols
            time.sleep(dit)
        # break between characters: 3 dits, 1 already made
        time.sleep(2*dit)
    elif c == ' ':
        # break between two words: 7 dits, (1+2)=3 already made
        time.sleep(4*dit)
    else:
        print(' Character ' + c + ' not in Morse code! ')

def valid_speed(speed):
    try:
        return int(speed) > 0 and int(speed) < 101
    except ValueError:
        return False

def setup_transmission_speed(dit):
    print ('\nSetting up speed of transmission to ' + str(round(1/dit)) + ' dits/s.\n')
    # send 5 pulses and 5 breaks of dit ms width to sync receiver
    for i in range(5):
        transmitter.set_laserpulse(dit)
        time.sleep(dit)
    # 0.5s break before transmission of data
    time.sleep(0.5)

print ('\n***** Morse Code Transmitter *****\n')

speed = input('Enter speed of transmission [1,100] dits/s: ')
while not valid_speed(speed):
    speed = input('Speed not valid, please reenter [1,100] dits/s: ')
dit = 1/int(speed)

# text to send via transmitter
```

```

text_out = input('\nEnter text to be transmitted: ')

# convert to uppercase
text_out = text_out.upper()

setup_transmission_speed(dit)

print ('***** Start of transmission *****\n')

index = 1
transmission_start = time.time()
for c in text_out:
    transmit_char(c)
    index = index +1
    # print character by character on screen
    sys.stdout.write(c)
    sys.stdout.flush()
    # new line on screen
    if index > 65 and c == ' ':
        index = 1
        print()
transmission_end = time.time()
print ('\n\n***** End of transmission *****\n')

print(str(len(text_out)) + ' characters transmitted in ' + str(round(transmission_end-
transmission_start,2)) + ' s.')

print('\n*****')

```

Listing 17: Morse\_Transmitter.py

### 7.6.4 Musterantworten

Aufgabe	Antwort
Aufgabe 1	<p>Datentyp Dictionary: zu jedem Schlüssel (Buchstabe aus dem Eingabealphabet) gibt es einen Wert (Liste mit Punkten und Strichen).</p> <p>Strich („dah“) dauert bei der Übertragung dreimal so lange wie Punkt (dit), deshalb wird der Strich durch „3“ und der Punkt durch „1“ repräsentiert.</p> <p>code = {'C':[3,1,3,1], 'D':[3,1,1]}</p>

Tabelle 44: Morse Transmitter Musterantworten



## 7.7 Morse Receiver

### 7.7.1 Lehrerinformation

Verwendetes Pidget	Optical Receiver
Alltagsbezug	Vgl. Kapitel 7.6.1
Motivation für die Aufgabe	Vgl. Kapitel 7.6.1
Lernziele	Vgl. Kapitel 7.6.1, zusätzlich: <ul style="list-style-type: none"> <li>Die Schüler sollen aus den Zeitpunkten von Ereignissen (Flankenwechsel beim Input-Pin) auf den Ablauf rückschließen können (Länge der Pulse und der Pausen) und diesen Zusammenhang in Software modellieren können.</li> </ul>
Schulstufe	WPG (Wahlpflichtgegenstand, 6.-8. Klasse)
Nötiges Vorwissen	Morsecodierung Listen (Dictionaries) Schleifen (for, while) Verzweigungen (if) Wertevergleich (<, >, =, !=) Einfache arithmetische Berechnungen (+, -, *, /, %) Modul time (sleep, localtime)
Bezug Kompetenzmodell	Inhaltsdimension „Praktische Informatik“ <ul style="list-style-type: none"> <li>Algorithmen, Datenstrukturen und Programmierung</li> </ul> Handlungsdimension „Anwenden und Gestalten“ <ul style="list-style-type: none"> <li>Aufgaben mit Mitteln der Informatik modellieren</li> <li>Algorithmen entwerfen, implementieren und testen</li> </ul> Reflektieren und Bewerten <ul style="list-style-type: none"> <li>Gezielt nach Programmfehlern suchen und diese korrigieren</li> </ul>
Unterrichtsmethode	Partnerarbeit, Plenum
Dauer	3 Doppelstunden
Anmerkungen	Diese Aufgabe ist im Vergleich zu Aufgabe 7.6.1 wesentlich komplexer, deshalb wird auch mehr Zeit dafür eingeplant. Die Schwierigkeit der Aufgabe besteht darin, aus den Zustandswechseln des GPIO Eingangs die Dits und Dahs und die Pausen zu extrahieren. Dazu werden in der Musterlösung die Zeiten der Zustandswechsel in die Variablen <code>rise_time</code> (steigende Flanke) und <code>fall_time</code> (fallende Flanke) gespeichert. Die Differenz zwischen <code>rise_time</code> und darauffolgender <code>fall_time</code> ist die Länge eines Laserpulses. Diese Länge durch die Länge eines Dits dividiert (und gerundet) ergibt 1 (für ein Dit) oder 3 (für ein Dah). Umgekehrt ergibt die Differenz aus der <code>fall_time</code> und der darauf folgenden <code>rise_time</code> die Länge der Pause. Diese Länge ebenfalls auf die „Einheitslänge“ Dit normiert ergibt 1 für eine Pause innerhalb eines Buchstabens, 3 für die Pause zwischen zwei Buchstaben und 7 für die Pause zwischen zwei Wörtern. Es sind in Bezug auf die Pausen in der Musterlösung niedrigere Grenzen verwendet worden, um timing-Instabilitäten auszugleichen: Alle Pausen länger als 1 Dit (also 2 Dits und mehr) sind eine Grenze zwischen zwei Buchstaben → der Buchstabe muss dekodiert und ausgegeben werden. Ist die Pause auch länger als 5 Dits,

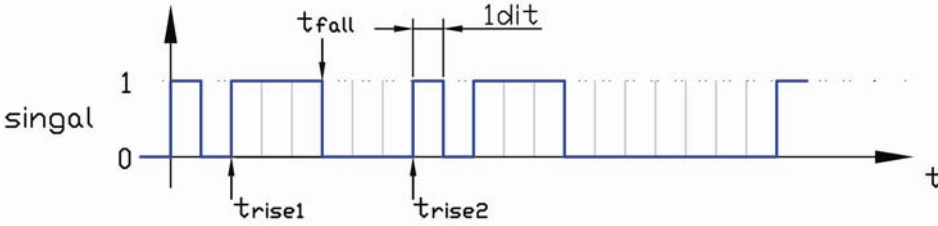
	<p>dann ist es eine Pause zwischen zwei Wörtern → es muss zusätzlich ein Leerzeichen ausgegeben werden.</p> <p>Das Erkennen von steigenden Flanken funktioniert beim Optical Receiver Pidget genauso wie bei den Pidgets mit einem Taster (Dice, 7 Segment Display) und ist in dieser Arbeit schon angeführt worden. Analog dazu funktioniert das Erkennen fallender Flanken. Das Ende der Übertragung wird über ein timeout erkannt. Wenn länger als 15 Dits keine steigende Flanke ankommt, wird das Empfängerprogramm beendet. Zuvor muss noch das letzte übertragene Zeichen ausgegeben werden. Dazu werden die Variablen</p> <pre>state = True prev_state = False</pre> <p>gesetzt. Damit wird dem Programm eine steigende Flanke simuliert und der Programmteil, der die Ausgabe übernimmt, noch einmal ausgeführt.</p> <p>Die Schüler sollen die Aufgabe im Zweierteam erledigen. Falls vor dieser Aufgabe der Morse Transmitter bearbeitet wurde, sollen die Teams nach Möglichkeit gleich bleiben.</p> <p>Am Ende von Teilaufgabe 2 ist es wichtig, dass die Ergebnisse der Schüler kontrolliert und ggf. verbessert oder abgeglichen werden. Das in Teilaufgabe 2 implementierte Programmstück ist der Rahmen für alle weiteren Aufgaben.</p> <p>Nach Teilaufgabe 4 ist es ebenfalls sinnvoll, die Ergebnisse der Schüler im Plenum zu diskutieren, denn diese Ergebnisse bilden die Basis für Teilaufgabe 5.</p> <p>Die restlichen Teilaufgaben können die Teams wieder selbständig bearbeiten.</p>
--	--

Tabelle 45: Morse Receiver Lehrerinformation

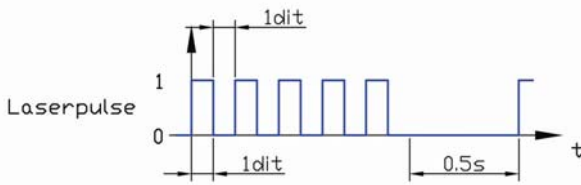
### 7.7.2 Arbeitsblatt

<b>Morse Receiver</b>	
<p>Deine Aufgabe ist es, den Optical Receiver so zu programmieren (<code>Morse_Receiver.py</code>), dass er morsecodierte Laserimpulse am Bildschirm ausgibt. Unser Ausgabealphabet besteht auf den folgenden Zeichen:</p> <p style="text-align: center;">A ... Z, 0 ... 9, ( ) + , - . / : ; = ? @</p> <p>Ungültige Morsecodes am optischen Eingang sollen gemeinsam mit einer Fehlermeldung am Bildschirm ausgegeben werden.</p>	
Aufgabe 1	<p>Das Programm muss den empfangenen Morsecode in Ausgabertext übersetzen. Diskutiere mit deinem Partner in welchem Datentyp du die Übersetzungstabelle zwischen Zeichen und Morsecode speichern würdest. Notiere das Ergebnis eurer Diskussion hier:</p> <p>Der Morsecode besteht aus zwei Symbolen, dem Punkt und dem Strich, die sich bei der Übertragung durch unterschiedliche Impulslängen unterscheiden. Wie würdet ihr im oben gewählten Datentyp den Punkt und den Strich repräsentieren?</p> <p>Punkt als _____ Strich als _____</p> <p>Wie würde eure Übersetzungstabelle ausschauen, wenn sie nur aus den Buchstaben C (Morsecode <code>-. .</code>) und D (<code>-..</code>) bestünde?</p> <p><code>reverse_code =</code></p>
Aufgabe 2	<p>Der Hauptteil des Programms wird in einer Schleife laufen, die erst nach dem letzten übertragenen Zeichen verlassen wird (<code>while not end_of_transmission</code>). Innerhalb der Schleife wird immer wieder der Zustand (<code>state</code>) des Optical Receivers abgefragt. Der <code>state</code> ist <code>True</code> (oder „1“), wenn ein optisches Signal anliegt, er ist <code>False</code> (oder „0“), wenn kein Signal anliegt.</p> <p>Damit dein Programm herausfinden kann, wann sich der Zustand des Optical Receivers geändert hat, muss es den Zustand speichern, den der Receiver im letzten Schleifendurchlauf hatte (<code>prev_state</code>).</p> <p>Wenn <code>prev_state</code> gleich <code>state</code> ist, hat sich nichts geändert. Der Receiver war entweder vorher auf „0“ und ist es jetzt auch noch, oder er war vorher auf „1“ und ist jetzt auch noch auf „1“.</p> <p>Welche Werte müssen <code>prev_state</code> und <code>state</code> haben, damit dein Programm daraus schließen kann, dass der Zustand des Receivers von „0“ auf „1“ gewechselt hat?</p> <p style="text-align: center;"><code>prev_state=_____ state=_____</code></p> <p>Umgekehrt: wie müssen <code>prev_state</code> und <code>state</code> sein, wenn der Receiver von „1“ auf „0“ gewechselt hat?</p> <p style="text-align: center;"><code>prev_state=_____ state=_____</code></p> <p>Dein Programm kann also durch Vergleich der Werte von <code>prev_state</code> und <code>state</code> herausfinden, ob das Signal von „0“ auf „1“ (oder umgekehrt) gesprungen ist. Diese Sprünge nennt man übrigens <i>Flanken</i>. Eine <i>steigende Flanke</i> führt von „0“ nach „1“, eine <i>fallende Flanke</i> von „1“ nach „0“.</p>

	<p>Beginne jetzt mit der Implementierung des Programms <code>Morse_Receiver.py</code>          Gehe folgender Maßen vor:</p> <ul style="list-style-type: none"> <li>• Importiere das Modul <code>time</code> (das brauchst du später)</li> <li>• Importiere das Modul für den Optical Receiver und lies dir die <code>description()</code> der Methoden des <code>Pidgets</code>. Welche Methode liefert dir den Zustand des Receivers? _____</li> <li>• Lege die Variablen <code>end_of_transmission</code>, <code>state</code> und <code>prev_state</code> an und initialisiere sie sinnvoll. Welchen Datentyp wählst du für die Variablen?</li> </ul> <table border="1" data-bbox="491 533 1329 674"> <thead> <tr> <th>Variable</th> <th>Typ</th> <th>Initialwert</th> </tr> </thead> <tbody> <tr> <td><code>end_of_transmission</code></td> <td></td> <td></td> </tr> <tr> <td><code>state</code></td> <td></td> <td></td> </tr> <tr> <td><code>prev_state</code></td> <td></td> <td></td> </tr> </tbody> </table> <ul style="list-style-type: none"> <li>• Implementiere die Schleife <code>while not end_of_transmission</code>, im Schleifenkörper sollen folgende Anweisungen stehen:             <ul style="list-style-type: none"> <li>○ <code>state</code> bekommt den aktuellen Wert des Receivers zugewiesen</li> <li>○ falls ein Übergang von „0“ auf „1“ war soll der Text „Steigende Flanke detektiert“ ausgegeben werden</li> <li>○ falls ein Übergang von „1“ auf „0“ war, soll der Text „Fallende Flanke detektiert“ ausgegeben werden.</li> <li>○ <code>prev_state</code> bekommt den Wert von <code>state</code> zugewiesen</li> </ul> </li> </ul> <p>Teste die Methode, in dem du Daten aus dem Optischen Transmitter mit ganz niedriger Übertragungsgeschwindigkeit (1Dit/s) zum Optischen Receiver überträgst. Pro Übergang zwischen Laser ein↔aus darf nur eine Zeile am Bildschirm ausgegeben werden, sonst ist dein Programm noch fehlerhaft!</p>	Variable	Typ	Initialwert	<code>end_of_transmission</code>			<code>state</code>			<code>prev_state</code>		
Variable	Typ	Initialwert											
<code>end_of_transmission</code>													
<code>state</code>													
<code>prev_state</code>													
<p>Aufgabe 3</p>	<p>Warten auf die Datenübertragung: Dein Programm soll nach dem Start auf die Daten vom Sender warten. Implementiere dazu vor der gerade implementierten Schleife ein Programmstück, das darauf wartet, dass der Zustand des Receivers auf „1“ geht. Dann soll das Programm erst fortgesetzt werden. Während des Wartens soll folgende Bildschirmausgabe zu sehen sein:</p> <pre> ***** Morse Code Receiver ***** Waiting for data transmission... ***** Start of Transmission *****     </pre> <p><b>Implementierungstipps:</b> Vermeide bei der Implementierung <i>busy waiting</i>, baue also an geeigneter Stelle ein <code>time.sleep(x)</code> ein. Wie lange darf oder soll <code>x</code> sein?</p> <p><code>x</code> darf/soll so lang sein:</p> <p>weil:</p> <p>Wenn der Wartestatus vorbei ist, hat das Programm die erste Flanke detektiert. Welchen Zustand müssen daher <code>prev_state</code> und <code>state</code> am Beginn der <code>while not end_of_transmission</code> Schleife haben? Weise ihnen diese Werte zu.</p> <p><code>prev_state=_____ state=_____</code></p>												

<p>Aufgabe 4</p>	 <p>Betrachte den Signalverlauf (blaue Linie) und versuche ihn als Morsecode aufzuschreiben: _____ Wofür steht der Code? _____</p> <p>Du hast die Umwandlung von Signal in Morsecode vermutlich so gemacht, dass du die Breite der Impulse betrachtet hast: Ist der Impuls ein „Kasterl“ breit, dann steht er für ein Dit, ist er drei „Kasterl“ breit, dann entspricht das einem Dah. Analog bei den Pausen.</p> <p>Beim Signalverlauf sind drei Zeitpunkte markiert. Wie kannst du die Dauer des Laserpulses aus <math>t_{rise1}</math>, <math>t_{fall}</math>, <math>t_{rise2}</math> ausrechnen?</p> <p><math>t_{pulse} =</math></p> <p>Wie lange ist die Dauer der darauffolgenden Pause?</p> <p><math>t_{pause} =</math></p> <p>Damit kennst du jetzt die Breite der Pulse und der Pausen. Du benötigst aber, wie viele „Kasterl“ <math>k</math> der Puls und die Pause breit sind. Wie kannst du das berechnen? Gib hier eine Formel an:</p> <p><math>k_{pulse} =</math> <span style="margin-left: 200px;"><math>k_{pause} =</math></span></p>
<p>Aufgabe 5</p>	<p>Genau dasselbe muss dein Programm machen. Bei einer steigenden Flanke soll der Zeitpunkt in die Variable <code>rise_time</code>, bei einer fallenden Flanke in die Variable <code>fall_time</code> gespeichert werden. Außerdem muss bei jedem Wechsel die Dauer des vorangegangenen Pulses / der vorangegangenen Pause als Vielfache von DIT berechnet werden.</p> <p>Für dein Programm: Überlege, wo die erste Flanke detektiert wird? Wie musst du <code>rise_time</code> und <code>fall_time</code> nach dieser Detektion initialisieren, damit dein Code funktioniert?</p> <p>Ersetze die Textausgaben „Steigende/Fallende Flanke detektiert“ durch ein Programmstück, das die im ersten Absatz beschriebene Funktion erfüllt.          Welche Werte erwartest du für die Breiten der Pulse? _____          Welche Werte erwartest du für die Breiten der Pausen? _____</p> <p>Lass dir die Berechnungsergebnisse am Bildschirm ausgeben.</p> <p>Teste dein Programm mit einem Morsecode von dem du die Abfolge der Puls- und Pausendauern kennst (vielleicht findet sich sogar ein passender Code auf diesem Arbeitsblatt...).</p> <p>Bemerkst du Abweichungen zu den erwarteten Ergebnissen?</p> <p>Was kannst du dagegen tun?</p>

<p>Aufgabe 6</p>	<p>Dein Programm hat jetzt zwei Bereiche, die relativ ähnlich sind: Es werden Flanken detektiert und es werden Zeitdauern als Vielfache von DIT berechnet. Ein Bereich behandelt die steigenden Flanken, der andere die fallenden. Welcher Bereich liefert die Breiten der Impulse? _____</p> <p>In diesem musst du nun den Schlüssel für unsere Übersetzungstabelle generieren. Zur Erinnerung: Ein Dictionary-Eintrag besteht aus einem Schlüssel und einem Wert. Der Schlüssel ist der übertragene Morsecode, der Wert das dazugehörige Zeichen aus unserem Ausgabealphabet.</p> <p>Der Schlüssel entsteht, indem einer Variable <code>received_code</code> die Ergebnisse der Berechnung („1“ oder „3“) angehängt werden.</p> <p>Welchen Datentyp wählst du für <code>received_code</code>?</p> <p>Wie initialisierst du die Variable?</p> <p>Implementiere nun das Programmstück, das (noch beliebig wachsende) Schlüssel generiert. Lass dir den Inhalt der Variable <code>received_code</code> am Bildschirm anzeigen.</p>
<p>Aufgabe 7</p>	<p>Ein Bereich kümmert sich um die Pulsbreiten, der andere um die Pausenlängen. In diesem musst du jetzt noch für die generierten Schlüssel den Wert holen und ausgeben. Dazu ist folgendes zu tun:</p> <ul style="list-style-type: none"> <li>• Wenn die Pause ein DIT lang ist, tue gar nichts.</li> <li>• Wenn die Pause 3 DITs lang ist, ist der Schlüssel vollständig übertragen.</li> <li>• Kontrolliere, ob der Schlüssel im Dictionary vorhanden ist             <ul style="list-style-type: none"> <li>◦ Falls ja: gib den Buchstaben aus, falls nein: gib eine Fehlermeldung aus</li> </ul> </li> <li>• Wenn die Pause 7 Dits lang ist (oder mehr als 5), gib zusätzlich ein Leerzeichen aus (Wort ist fertig übertragen).</li> <li>• Lösche den Inhalt der Variable <code>received_code</code></li> </ul> <p>Für die zeichenweise Ausgabe am Bildschirm kannst du folgende Befehle verwenden:</p> <pre>import sys  sys.stdout.write(c) sys.stdout.flush()</pre>
<p>Aufgabe 8</p>	<p>Damit bei längeren Texten sinnvolle Zeilenumbrüche entstehen, erweitere deine Textausgabe so, dass beim ersten Leerzeichen ab dem 65. Zeichen einer Zeile, ein Zeilenumbruch gemacht wird.</p>
<p>Aufgabe 9</p>	<p>Erweitere dein Programm so, dass der Empfang von Morsezeichen beendet wird, wenn länger als 15 DITs keine steigende Flanke detektiert wurde.</p> <p>Zum Beenden des Programms muss <code>end_of_transmission</code> auf <code>True</code> gesetzt werden. Allerdings befinden sich zu diesem Zeitpunkt noch Morsezeichen in der Variable <code>received_code</code>, daher musst du dafür sorgen, dass dein Programm den Inhalt der Variable noch in das Ausgabealphabet übersetzt und am Bildschirm ausgibt.</p>

<p>Bonus 1</p>	<p>Ein vollständig implementierter Morse Code Transmitter sendet vor den zu übertragenden Daten eine Sequenz, aus der der Empfänger die Übertragungsgeschwindigkeit (DITS/s) eruiieren kann.</p>  <p>Die Sequenz besteht aus 5 Pulsen und 5 Pausen, die je eine Länge von einem DIT haben. Dann folgt eine Pause von 0,5s. Implementiere eine Methode, die die Gesamtdauer der 5 Impulse misst und daraus die Dauer eines DITs errechnet. Dieser Wert soll dann für die Empfangsgeschwindigkeit beim Empfänger verwendet werden.</p> <p>Die Bildschirmausgabe soll so aussehen:</p> <pre> ***** Morse Code Receiver ***** Waiting for synchronization... Synchronization in progress... Synchronization completed Receiving at 100 dits/s. ***** Start of Transmission ***** DAS IST EIN PROBETEXT </pre>
<p>Bonus 2</p>	<p>Zusammenfassung: Erweitere dein Programm so, dass am Ende der Übertragung die Anzahl der empfangenen Zeichen (des Ausgabealphabets) und die Übertragungsdauer angezeigt werden. Die Ausgabe soll so aussehen:</p> <pre> ***** End of Transmission ***** 21 characters received in 1.64 s. ***** </pre> <p>Falls die Angabe (vor allem der Zeit) vom Sender abweicht: Überlege, wann für den Sender und wann für den Empfänger die Übertragung endet. Stichwort: timeout beim Empfänger.</p>

### 7.7.3 Musterlösung

```
#!/usr/bin/env python

import time
import sys
import Optical_Receiver as receiver

receiver.init()
test_mode = False

# Dictionary of reverse Morse codes for: ( ) + , - . / 0 ... 9 : ; = ? @ A ... Z
reverse_code = {'31331': '(', '313313': ')', '13131': '+', '331133': ',', '311113': '-',
                '131313': '.', '31131': '/', '33333': '0', '13333': '1', '11333': '2', '11133': '3', '11113': '4', '111
11': '5', '31111': '6', '33111': '7', '33311': '8', '33331': '9', '333111': ':', '313131': ';', '31113': '
=' , '113311': '?', '133131': '@', '13': 'A', '3111': 'B', '3131': 'C', '311': 'D', '1': 'E', '1131': 'F', '3
31': 'G', '1111': 'H', '11': 'I', '1333': 'J', '313': 'K', '1311': 'L', '33': 'M', '31': 'N', '333': 'O', '13
31': 'P', '3313': 'Q', '131': 'R', '111': 'S', '3': 'T', '113': 'U', '1113': 'V', '133': 'W', '3113': 'X', '3
133': 'Y', '3311': 'Z'}

def get_DIT_duration():
    state = True
    prev_state = True
    slopes = 1

    # waiting for first rising slope
    while not receiver.get_state():
        time.sleep(0.0001)

    synchronization_start = time.time()

    print ('Synchronization in progress...')

    # start of synchronization
    while slopes < 10:
        state = receiver.get_state()

        # rising slope
        if (state and not prev_state):
            slopes = slopes + 1

        # falling slope
        elif (not state and prev_state):
            slopes = slopes + 1

        prev_state = state

    synchronization_end = time.time()

    print ('\nSynchronization completed')
    return (synchronization_end - synchronization_start) / (slopes - 1)

print ('***** Morse Code Receiver *****')
print ('\nWaiting for synchronization...\n')

if test_mode:
    DIT = 0.02
else:
    DIT = get_DIT_duration()

print ('\nReceiving at ' + str(round(1/DIT)) + ' dits/s.')
print ('\n***** Start of Transmission *****\n')

time.sleep(0.5)

# waiting for first rising slope of data
```



```

while not receiver.get_state():
    time.sleep(0.0001)

rise_time = time.time()
fall_time = time.time()
transmission_start = time.time()
received_code = ''
end_of_transmission = False
state = True
prev_state = True
print_index = 1
char_received = 0

while not end_of_transmission:
    # transmission ends, when break is longer than 15 DITs
    if not state and (time.time() - fall_time) > (15*DIT) and not test_mode:
        end_of_transmission = True
        # print last received letter
        state = True
        prev_state = False

    # rising slope detected, calculate length of break, print character
    if (state and not prev_state):
        rise_time = time.time()
        length = rise_time - fall_time

        # break between two letters or two words
        if round(length/DIT) > 1:
            if received_code in reverse_code:
                char_received = char_received + 1
                sys.stdout.write(reverse_code[received_code])
                sys.stdout.flush()
                print_index = print_index + 1

            # break between two words print space character
            if round(length/DIT) > 5:
                char_received = char_received + 1
                sys.stdout.write(' ')
                sys.stdout.flush()
                print_index = print_index + 1

            # new line
            if print_index > 65:
                print()
                print_index = 1
        else:
            print('\nTransmission Error: ' + received_code + ' not in Morse code!')
            print_index = 1
            received_code = ''

    # falling slope detected, calculate length of pulse, add symbol to received_code
    elif (not state and prev_state):
        fall_time = time.time()
        length = fall_time - rise_time

        # add dit or dah to received code
        if round(length/DIT) == 1:
            received_code = received_code + '1'
        else:
            received_code = received_code + '3'

    # reduce CPU load
    time.sleep(0.001)
    prev_state = state
    state = receiver.get_state()

# transmission ended 15-3 = 12 DITs before now

```

```

transmission_end = time.time() - 12 * DIT

print ('\n\n***** End of Transmission *****\n')
print (str(char_received-1) + ' characters received in ' + str(round(transmission_end -
transmission_start,2)) + ' s.')
print ('\n*****')

```

Listing 18: Morse\_Receiver.py

### 7.7.4 Musterantworten

Aufgabe	Antwort												
Aufgabe 1	<p>Datentyp Dictionary: Zu jedem Schlüssel (Liste mit Punkten und Strichen) gibt es einen Wert (Buchstabe aus dem Ausgabealphabet).</p> <p>Strich („dah“) dauert am optischen Eingang dreimal so lange wie Punkt (dit), deshalb wird der Strich durch „3“ und der Punkt durch „1“ repräsentiert.</p> <pre>reverse_code = {'3131':'C', '311':'D'}</pre>												
Aufgabe 2	<p>prev_state = False    state = True    für eine steigende Flanke  prev_state = True     state = False   für eine fallende Flanke</p> <p>get_state() liefert den aktuellen Zustand des Receivers</p> <table border="1"> <thead> <tr> <th>Variable</th> <th>Typ</th> <th>Initialwert</th> </tr> </thead> <tbody> <tr> <td>end_of_transmission</td> <td>Boolean</td> <td>False</td> </tr> <tr> <td>state</td> <td>Boolean</td> <td>False</td> </tr> <tr> <td>prev_state</td> <td>Boolean</td> <td>False</td> </tr> </tbody> </table>	Variable	Typ	Initialwert	end_of_transmission	Boolean	False	state	Boolean	False	prev_state	Boolean	False
Variable	Typ	Initialwert											
end_of_transmission	Boolean	False											
state	Boolean	False											
prev_state	Boolean	False											
Aufgabe 3	<p>Die Wartezeit x soll klein sein (max. 1% von DIT), weil diese Zeit in den Messfehler der ersten Zeitnehmung eingeht. x soll nur so groß sein, dass die CPU nicht völlig ausgelastet ist.</p> <p>Nach der ersten Flanke muss der Wert der beiden Statusvariablen auf True sein.</p>												
Aufgabe 4	<p>Der Code ist <code>· - · -</code> und steht für AA[Leerzeichen]</p> <pre> t_pulse = t_fall - t_rise1 t_pause = t_rise2 - t_fall  k_pulse = t_pulse/dit k_pause = t_pause/dit </pre>												
Aufgabe 5	<p>Erwarteter Wert für die Pulse: 1 und 3 Dits  Erwarteter Wert für die Pausen: 1, 3 und 7 Dits</p> <p>Abweichungen: die Rechenergebnisse haben Kommastellen  Abhilfe: nicht in einen Integer casten, sondern runden.</p>												
Aufgabe 6	<p>Die Impulsbreiten (und damit die Dits und Dahs) werden nach Detektion einer fallenden Flanke behandelt.</p> <p>received_code ist ein String.</p> <pre>received_code = ''</pre>												

Tabelle 46: Morse Receiver Musterantworten

## 8 Rück- und Ausblick

Rückblickend kann aus Sicht des Autors gesagt werden, dass viele persönliche Erwartungen an diese Arbeit erfüllt worden sind. Es hat Spaß gemacht mit dem Raspberry Pi zu arbeiten, es war interessant die Pidgets zu entwerfen und es war lehrreich in einer für den Autor neuen Programmiersprache die Aufgaben zu entwickeln und zu lösen.

Die Arbeit als Ganzes betrachtet, ist es in der Einschätzung des Autors gelungen, einem Leser alle notwendigen Informationen zu geben, um am Raspberry Pi samt Pidget die entwickelten Aufgaben im Unterricht bearbeiten zu lassen. Das in einer Vorbesprechung zu dieser Arbeit einmal lapidar formulierte Ziel „*Man soll das Gesamtkunstwerk einem Kollegen geben können und der kann das dann unterrichten.*“ scheint dem Autor als erreicht.

Spannend bleibt, wie sich die Ergebnisse dieser Arbeit in konkreten Unterrichtssituationen bewähren. Hier bleibt auch die Frage offen, in wie weit die ohnehin nur wenigen Schülerinnen im Wahlpflichtgegenstand Informatik mit den doch sehr technischen Pidgets erreicht und für die Aufgaben begeistert werden können.

Das Konzept, die Pidgets sehr einfach zu halten, scheint auch mit dem heutigen Wissensstand noch sinnvoll. Wünschenswert wäre allerdings, dass eine nächste Generation mehr Eingabemöglichkeiten (Schalter, Taster) hat.

Für die Zukunft kann gesagt werden, dass es spannend bleibt, welche Projekte die Mitglieder der weltweiten Community rund um den Raspberry Pi noch hervorbringen. Diese Projekte können immer wieder wertvolle Anregungen für den eigenen Informatikunterricht bringen.

Die Raspberry Pi Foundation entwickelt laufend Erweiterungen für den Raspberry Pi, so wurde vor kurzem ein Kameramodul für den Raspberry Pi veröffentlicht, das an das Camera Serial Interface angeschlossen werden kann (vgl. Abbildung 1 und Tabelle 1).

Die Firma Playtastic bietet einen Roboterarm [35] mit Kabelfernbedienung an. Dieser Arm könnte ähnlich wie die Fernsteuerung des Pi-Car umgebaut werden, und damit ließe sich der Arm dann mit dem Raspberry Pi ansteuern.

Für iOS Geräte gibt es das App Ardumote [36], das dazu gedacht ist Arduinos über WLAN fernzusteuern. Es existiert aber auch ein Python Script, das mit diesem App kommuniziert, und damit lassen sich Pidgets oder andere Dinge, die am GPIO Port angeschlossen sind vom iOS Gerät steuern. Das App kostet 3,59€ für iPhone und 5,49€ für iPad.

Der letzte Satz dieser Arbeit ist ein Zitat von der Homepage der Raspberry Pi Foundation:

Tip of the day: Make something from scratch at least once a week -- a meal; a catapult; a program; a rocket; an origami frog. It doesn't matter what. [37]

## 9 Quellenverzeichnis

- [1] Raspberry Pi Foundation, "About us," [Online]. Available: <http://www.raspberrypi.org/about>. [Accessed 26 06 2013].
- [2] "Python Programming Language," [Online]. Available: <http://www.python.org/>. [Accessed 29 06 2013].
- [3] "Codecademy," [Online]. Available: <http://www.codecademy.com>. [Accessed 27 06 2013].
- [4] J. C. Maxwell, Five of Maxwell's Papers, Create Space Independent Publishing Platform, 2011.
- [5] „Raspberry Pi,“ [Online]. Available: [http://de.wikipedia.org/wiki/Raspberry\\_Pi](http://de.wikipedia.org/wiki/Raspberry_Pi). [Zugriff am 13 05 2013].
- [6] E. Engelhardt, Coole Projekte mit Raspberry Pi, München: Franzis, 2013.
- [7] „RS Components,“ [Online]. Available: [http://at.rs-online.com/web/c/?searchTerm=raspberry-pi-products\\_at&searchType=Offers](http://at.rs-online.com/web/c/?searchTerm=raspberry-pi-products_at&searchType=Offers). [Zugriff am 19 06 2013].
- [8] „RPi Hardware History,“ [Online]. Available: [http://elinux.org/RPi\\_HardwareHistory](http://elinux.org/RPi_HardwareHistory). [Zugriff am 13 05 2013].
- [9] "Raspberry Pi," [Online]. Available: <http://www.raspberrypi.org/downloads>. [Accessed 19 05 2013].
- [10] "RaspBMC Download," [Online]. Available: <http://www.raspbmc.com/download/>. [Accessed 13 05 2013].
- [11] "RazDroid," [Online]. Available: <http://blog.pi3g.com/tag/razdroid/>. [Accessed 13 05 2013].
- [12] "RPi Distributions," [Online]. Available: [http://elinux.org/RPi\\_Distributions](http://elinux.org/RPi_Distributions). [Accessed 13 05 2013].
- [13] A. Beug, "alex's coding playground," [Online]. Available: <http://www.alexpage.de/>. [Accessed 04 06 2013].
- [14] "PuTTY," [Online]. Available: <http://www.putty.org/>. [Accessed 14 06 2013].
- [15] "MobaXterm," [Online]. Available: <http://mobaxterm.mobatek.net/>. [Accessed 14 06 2013].
- [16] „WinSCP,“ [Online]. Available: <http://winscp.net/eng/docs/lang:de>. [Zugriff am 14 06 2013].

- [17] "Automate file transfers," [Online]. Available: [http://winscp.net/eng/docs/guide\\_automation](http://winscp.net/eng/docs/guide_automation). [Accessed 14 06 2013].
- [18] "Alltheware," [Online]. Available: <http://alltheware.wordpress.com/2012/12/11/easiest-way-sd-card-setup/>. [Accessed 15 06 2013].
- [19] "XQuartz," [Online]. Available: <http://xquartz.macosforge.org/landing/>. [Accessed 15 06 2013].
- [20] "Mike Cook's Magic Wand," [Online]. Available: <http://www.raspberrypi.org/archives/1663>. [Accessed 20 06 2013].
- [21] Adafruit, "Adafruit LCD + Keypad," [Online]. Available: <http://www.adafruit.com/products/1110>. [Accessed 29 06 2013].
- [22] eLinux.org, "RPi Low-level peripherals," [Online]. Available: [http://elinux.org/RPi\\_Low-level\\_peripherals](http://elinux.org/RPi_Low-level_peripherals). [Accessed 29 06 2013].
- [23] G. v. Loo, "GPIO Pads Control2," [Online]. Available: <http://de.scribd.com/doc/101830961/GPIO-Pads-Control2>. [Accessed 13 05 2013].
- [24] Sparkfun, "Logic Level Converter," [Online]. Available: <https://www.sparkfun.com/products/8745>. [Accessed 31 05 2013].
- [25] Pi Cars, [Online]. Available: <http://pi-cars.com/>. [Accessed 31 05 2013].
- [26] „cymplecy,“ [Online]. Available: <http://cymplecy.wordpress.com/>. [Zugriff am 31 05 2013].
- [27] Conrad Electronic, [Online]. Available: <http://www.conrad.at/ce/de/product/776265/>. [Zugriff am 19 06 2013].
- [28] Allgemeine Unfallversicherungsanstalt (AUVA), „M080 Grundlagen der Lasersicherheit,“ Wien, 2009.
- [29] „Python Package Index,“ [Online]. Available: <https://pypi.python.org/pypi/RPi.GPIO>. [Zugriff am 25 05 2013].
- [30] Google, "Google code," [Online]. Available: <http://code.google.com/p/pidgets/source/browse/>. [Accessed 29 06 2013].
- [31] Bundesministerium für Unterricht, Kunst und Kultur, „Informatik Leitfaden zur kompetenzorientierten Reifeprüfung an AHS,“ Wien, 2013.
- [32] „Pulsweitenmodulation,“ [Online]. Available: <http://de.wikipedia.org/wiki/Pulsweitenmodulation>. [Zugriff am 17 06 2013].
- [33] „Morsecode - Wikipedia,“ [Online]. Available: [de.wikipedia.org/wiki/Morsecode](http://de.wikipedia.org/wiki/Morsecode). [Zugriff am 24 06 2013].

- [34] „ZDFheute\_Musik - Youtube,“ [Online]. Available: <https://www.youtube.com/watch?v=bLbMw9LwJdE>. [Zugriff am 24 06 2013].
- [35] Playtastic, „Playtastic Roboterarm,“ [Online]. Available: <http://www.playtastic.de/Roboter-Arm-NC-1424-919.shtml>. [Zugriff am 28 06 2013].
- [36] Samrat Amin, [Online]. Available: <http://samratamin.com/Arduote.html>. [Zugriff am 28 06 2013].
- [37] Raspberry Pi Foundation, "Raspberry Pi," [Online]. Available: <http://www.raspberrypi.org/>. [Accessed 27 06 2013].

## 10 Abbildungsverzeichnis

Alle Abbildungen dieser Arbeit stammen vom Autor.

Abbildung 1: Vorder- und Rückseite des Raspberry Pi .....	8
Abbildung 2: Raspi-config Programm .....	16
Abbildung 3: Synaptic Paketverwaltung .....	19
Abbildung 4: Pakete herunterladen .....	19
Abbildung 5: IDE Geany .....	20
Abbildung 6: Pakete vormerken .....	21
Abbildung 7: Benutzer hinzufügen .....	23
Abbildung 8: USB Image Tool.....	24
Abbildung 9: PuTTY .....	25
Abbildung 10: PuTTY Configuration .....	25
Abbildung 11: MobaXterm .....	26
Abbildung 12: MobaXterm Session settings.....	26
Abbildung 13: WinSCP Anmeldung .....	27
Abbildung 14: WinSCP .....	28
Abbildung 15: Mac OS Terminal .....	29
Abbildung 16: X11 Einstellungen .....	29
Abbildung 17: Freigabe unter Mac OS .....	30
Abbildung 18: GPIO Port des Raspberry Pi.....	35
Abbildung 19: Pin-Belegung GPIO, Bestückungsseite.....	36
Abbildung 20: Pin-Belegung GPIO, Lötseite.....	36
Abbildung 21: Schaltungsvariante .....	39
Abbildung 22: Spannungsteiler .....	39
Abbildung 23: Schaltplan Pi-Car.....	40
Abbildung 24: Fernsteuerung geschlossen - geöffnet .....	41
Abbildung 25: Kabel angeschlossen .....	41
Abbildung 26: Fertig umgebaute Fernbedienung.....	42
Abbildung 27: Außen- und Innenansicht LED Strip.....	42
Abbildung 28: Schaltplan LED Strip .....	43
Abbildung 29: Außen-, Innen- und Rückansicht 7-Segment Display.....	43
Abbildung 30: Schaltplan 7-Segment Display .....	44
Abbildung 31: Außen- und Innenansicht Optical Transmitter .....	45
Abbildung 32: Schaltplan Optical Transmitter.....	46
Abbildung 33: Außen- und Innenansicht Optical Receiver .....	47
Abbildung 34: Schaltplan Optical Receiver .....	47
Abbildung 35: Digitalservo .....	48
Abbildung 36: Impulsdiagramm.....	48
Abbildung 37: Ansteuerung Digitalservo.....	49
Abbildung 38: Außen- und Innenansicht Dice .....	49
Abbildung 39: Schaltplan Dice .....	50
Abbildung 40: Bauteile und Werkzeuge .....	51
Abbildung 41: Vollständiges Timingdiagramm.....	85

## 11 Tabellen

Tabelle 1: Anschlüsse und Komponenten des Raspberry Pi .....	8
Tabelle 2: Vergleich Raspberry Pi Modelle .....	9
Tabelle 3: Zuordnung Revision zu Modell.....	9
Tabelle 4: Zusatzkomponenten.....	10
Tabelle 5: Kosten bei Betrieb als PC.....	11
Tabelle 6: Kosten bei Zugriff über XTerminal (WLAN).....	11
Tabelle 7: Kosten bei Zugriff über XTerminal (LAN).....	12
Tabelle 8: Interaktionsausprägungen.....	31
Tabelle 9: Charakterausprägungen .....	31
Tabelle 10: Fachzugehörigkeitsausprägungen.....	31
Tabelle 11: Kommunikationsausprägungen .....	32
Tabelle 12: Realisierbarkeitsausprägungen .....	32
Tabelle 13: Verständlichkeitsausprägungen.....	33
Tabelle 14: Initialzustände der GPIO Pins .....	38
Tabelle 15: Mögliche Gestaltungsfehler.....	38
Tabelle 16: Klassifikation Pi-Car .....	40
Tabelle 17: Stückliste Pi-Car .....	41
Tabelle 18: Klassifikation LED Strip .....	42
Tabelle 19: Stückliste LED Strip.....	43
Tabelle 20: Klassifikation 7-Segment Display.....	44
Tabelle 21: Stückliste 7-Segment Display.....	44
Tabelle 22: Zuordnung GPIO Pins zu Treiber IC.....	45
Tabelle 23: Klassifikation Optical Transmitter .....	45
Tabelle 24: Stückliste Optical Transmitter.....	46
Tabelle 25: Klassifikation Optical Receiver.....	47
Tabelle 26: Stückliste Optical Receiver .....	48
Tabelle 27: Klassifikation Dice .....	50
Tabelle 28: Stückliste Dice.....	51
Tabelle 29: Arbeitsschritte Pidgetaufbau .....	53
Tabelle 30: Arbeitsschritte Kabelherstellung .....	54
Tabelle 31: Lauflicht Lehrerinformation.....	63
Tabelle 32: Lauflicht Musterantworten .....	67
Tabelle 33: Divisionen in Python 2.x und 3.x .....	68
Tabelle 34: Binärzahlendarstellung Lehrerinformation .....	69
Tabelle 35: Binärzahlendarstellung Musterantworten.....	70
Tabelle 36: Binary Battle Lehrerinformation .....	72
Tabelle 37: Binary Battle Musterantworten .....	76
Tabelle 38: Black Jack Lehrerinformation .....	78
Tabelle 39: Gewinnwahrscheinlichkeit Computer .....	84
Tabelle 40: Black Jack Musterantworten .....	85
Tabelle 41: Clock Lehrerinformation.....	86
Tabelle 42: Clock Musterantworten .....	90
Tabelle 43: Morse Transmitter Lehrerinformation.....	92
Tabelle 44: Morse Transmitter Musterantworten.....	96
Tabelle 45: Morse Receiver Lehrerinformation .....	98
Tabelle 46: Morse Receiver Musterantworten .....	106



## 12 Listings

Listing 1: Auflistung der Datenträger des Systems .....	15
Listing 2: Auflistung der Datenträger des Systems (inkl. SD-Karte).....	15
Listing 3: Nutzung von Dice.py .....	55
Listing 4: Dice.py .....	56
Listing 5: LED_Strip.py .....	58
Listing 6: Seven_Segment_Display.py .....	59
Listing 7: Optical_Transmitter.py .....	60
Listing 8: Optical_Receiver.py .....	61
Listing 9: Light_Chaser.py .....	66
Listing 10: Methode set_binary(number) .....	70
Listing 11: Binary_DisplayTest.py .....	70
Listing 12: Binary_Battle.py .....	76
Listing 13: Black_Jack.py .....	83
Listing 14: Black_Jack_Test.py .....	83
Listing 15: Clock.py .....	89
Listing 16: Switch_Test.py .....	89
Listing 17: Morse_Transmitter.py .....	96
Listing 18: Morse_Receiver.py .....	106

## 13 Danksagungen

Zuerst möchte ich mich bei a.Univ.-Prof. Dipl.-Ing. Dr. Günther Blaschek bedanken, dass er die Betreuung dieser Arbeit übernommen hat. Er hat großen Anteil daran, dass diese Diplomarbeit von Beginn an einen geordneten Verlauf genommen hat.

Natürlich ist es relativ einfach, grundsätzlich sehr sehr viele Füllwörter in eigentlich relativ wenig Inhalt zu verpacken. Ich bedanke mich bei meinen beiden Reviewern Ana (das ist kein Tippfehler!) und David Mariacher, dass das nicht passiert ist.

Danke an meine Studienkollegen, besonders an Günter Öller, der große Teile der vergangenen acht Semester gemeinsam mit mir bestritten hat.

Ich danke Josef Burgstaller und Gerald Fattinger für die Laserbearbeitung der Pidget Frontplatten, sowie Siegfried Humer für den elektronischen Beistand.

Danke an meine Schwiegereltern Maria (Diplomandenmensa) und Josef (Pidget Holzgehäuse) Wurzinger.

Ich danke meinen Eltern für so vieles. Unter anderem auch für das geduldige Warten auf einen Studienabschluss.

Und DANKE an meine liebe Frau Aurelia für ... alles.

## 14 Lebenslauf

### Profil

Name: Klaus Rabeder  
 Geboren: 25. April 1969  
 Familienstand: Verheiratet

### Bildung / Ausbildung

1975 - 1983 VS und HS Feldkirchen  
 1983 - 1988 HTL Maschinenbau, Linz. Abschluss mit gutem Erfolg  
 1988 - 1990 4 Semester Studium Toningenieur an der TU Graz und der Hochschule für Musik und Darstellende Kunst Graz. Kein Abschluss  
 2003 - 2010 nebenberufliches Studium Technische Informatik an der Johannes Kepler Universität Linz. Abschluss Bakk.techn.  
 2009 - Studium Lehramt für Informatik und Physik. Geplanter Abschluss mit Ende Sommersemester 2013.  
 2010 Leistungs- und Förderstipendium der Technisch - Naturwissenschaftlichen Fakultät der Johannes Kepler Universität.

### Berufserfahrung

1991 - 2001 **Institut für Halbleiterphysik, Johannes Kepler Universität Linz**  
 Institutstechniker: technische Betreuung des Reinraums, Betrieb einer Hochvakuumaufdampfanlage und eines reaktiven Ionenäzters, Strahlenschutzbeauftragter.  
 Betreuung des Institutsservers und der Institutshomepage.

1999 - 2001 **BBRZ Linz**  
 Freier Mitarbeiter: Unterricht von PC-Hardware- und Softwaretechnik.

2001 - 2009 **E+E Elektronik, Engerwitzdorf**  
 Prozesstechniker für Schutzbelackung elektronischer Systeme und für Ultraschallschweißen.  
 Gruppenleiter Wartung: Wartung von Produktionsmaschinen zum Fertigen von Dünnschichtsensoren, Realisierung von Automatisierungsprojekten  
 Prozesstechniker für Lasertrimmung von Strömungssensoren.

### Hobbys

Musik Tuba, E-Bass, Kontrabass; Obmann der musik.feldkirchendonau

## **15 Eidesstattliche Erklärung**

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe. Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Linz, Juli 2013

Klaus Rabeder